Modélisation et gestion d'univers 3D

Gérard Hégron, Patrick Rives

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports Techniques

N°61

MODÉLISATION ET GESTION D'UNIVERS 3D

Gérard HEGRON
Patrick RIVES

Décembre 1985

Gérard HÉGRON

Patrick RIVES

MODELISATION ET GESTION D'UNIVERS 3D

Une première approche à partir du logiciel PADL-2

Publication interne
n° 269

Octobre 1985

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 - RENNES CÉDEX
FRANCE
Tél. : (99) 36.20.00
Télex : UNIRISA 95 0473 F

Publication Interne n° 269 - Octobre 85

60 pages

MODELISATION ET GESTION D'UNIVERS 3D

(UNE PREMIÈRE APPROCHE À PARTIR DU LOGICIEL PADL-2)

GÉRARD HEGRON

PATRICK RIVES

Résumé. - Les systèmes de CAO classiques comme EUCLID, CATIA et PADL-2, ont été surtout conçus pour étudier des objets uniques et statiques. Dans la perspective du développement d'un logiciel de description et de modélisation de scènes 3D qui autorisera la gestion d'objets mobiles et de l'univers local perçu par un observateur en mouvement. Nous présentons dans ce rapport une première approche autour du logiciel PADL-2. Après une description générale de PADL-2 (structures générales, représentations internes des objets, fonctionnalités des modules), nous faisons la revue des diverses modifications, extensions et utilisations du système envisageables. Cette évaluation de PADL-2 nous a conduits à réaliser plusieurs extensions : ajout de nouvelles commandes au langage, intégration du mouvement, et développement d'un logiciel simple de description d'univers 3D. Avant de décrire la façon d'utiliser le système, un algorithme de gestion dynamique de la scène en fonction du déplacement de l'observateur utilisant un graphe de connexité spatiale des objets est également présenté.

Abstract. - CAD/CAM systems like EUCLID, CATIA and PALD-2, have been overall conceived to design static and single objects. In order to develop a 3D scene modelling system which will allow the management of moving objects and of the local universe perceived by an observer, we present in this apper our first developments realized from the geometric modelling system PADL-2. After a general description of PADL-2 (the main features, the geometric models of the objects, the use of the packages). We list the different modifications, extensions and uses we can do about this system. This valuation of PADL-2 led us to implement several extensions : addition of new commands in the language, integration of the motion, and development of a simple modelling system of 3D universe. Before describing how to use PADL-2, an algorithm of the dynamic management of the scene according to the motion of an observer based on spatial neighbourhood properties of the objects, is presented.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L.A. 227)
UNIVERSITÉ DE RENNES 1 I.N.S.A. DE RENNES

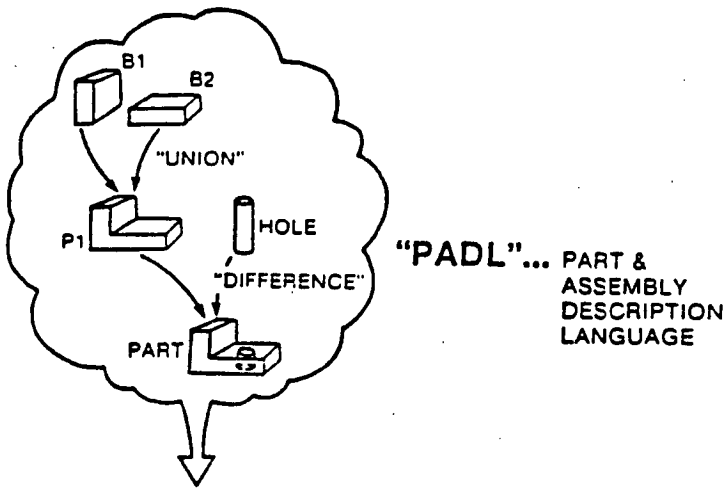
INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

MODELISATION ET GESTION D'UNIVERS 3D

(Une première approche à partir du logiciel PADL-2)

I. INTRODUCTION

PADL (Part and Assembly Description Language) est un système de modélisation géométrique d'objets solides tridimensionnels, développé à l'Université de Rochester (U.S.A). Le langage permet de construire les objets par un ensemble de combinaisons géométriques (union, intersection, différence) de primitives (parallélépipèdes, cylindres, sphères, cônes, tores,...) dont on spécifie les dimensions (hauteur, largeur, rayon,...) et la position dans l'espace (voir figure 1.1.).



PADL-2

```
B1 = BLO(..)AT(..)
B2 = BLO(..)AT(..)
P1 = B1 UN B2
HOLE = CYL(..)AT(..)
PART = P1 DIF HOLE
DISP PART
```

Figure 1.1: Description d'un objet avec PADL-2

Ce modèle de représentation des objets, appelé méthode CSG ("Constructive Solid Geometry"), est l'un des quatre modèles générés par le système, mais le seul qui est accessible à l'utilisateur.

L'opérateur dialogue avec le système via un interpréteur (voir Fig.1.2). Les instructions du langage sont entrées au clavier sous une forme littérale (texte) en mode interactif et sont de deux types:

. définitions : elles permettent de créer les objets (voir les 5 premières instructions de la figure 1.1);

. commandes: elles provoquent l'exécution d'un programme d'application (choix du périphérique, mise à jour des paramètres de visualisation, affichage,...) (sur la figure 1.1, la commande DISP permet d'afficher l'objet PART).

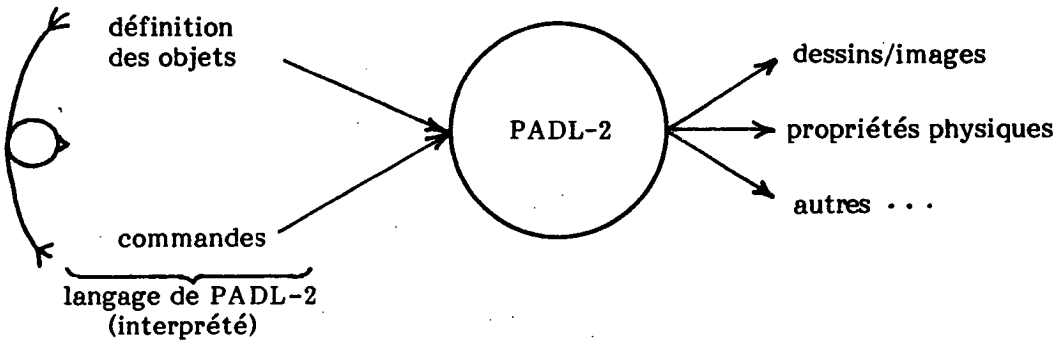


Figure 1.2: Configuration générale du système

Le lecteur trouvera une description complète du langage dans le Manuel Utilisateur de PADL-2 (document UM-10/1-2 "PADL-2 USERS MANUAL").

L'objet décrit par l'opérateur peut être visualisé de différentes manières: soit sous la forme d'un dessin au trait avec ou sans élimination des lignes cachées, soit sous la forme d'une image comportant des ombrages avec différents niveaux d'intensité. Plusieurs vues de la scène sont obtenues en modifiant la position du point de vue, le champ d'observation, la position des sources lumineuses, la couleur des objets, etc.. L'utilisateur peut également avoir des informations quantitatives sur les propriétés physiques des objets.

PADL-2 est avant tout un système "ouvert" destiné à évoluer et à s'intégrer dans des applications diverses. Plusieurs projets sont actuellement à l'étude à l'IRISA:

.réalisation complète d'une chaine de simulation d'un robot mobile possédant un capteur d'images évoluant dans un environnement non déterministe (projet d'étude

commun entre l'IRISA et THOMSON-LER [1], [2]). Un niveau supérieur de description de scènes 3D est adjoint à PADL-2 pour mobiliser l'Univers 3D dans lequel évolue le robot ;

.intégration de PADL-2 dans un système de simulation des interactions capteur-environnement dans l'espace 3D [3], [4] ;

A cet effet nous développons dans ce qui suit:

*une description du système, des différents modèles de représentation interne des objets et des divers modules de PADL-2 ;

*les évolutions possibles du système et la façon de les mettre en oeuvre ;

*les extensions réalisées à l'IRISA, en particulier les nouvelles commandes ajoutées au langage, le logiciel de description de scènes 3D, et un algorithme de gestion dynamique de la visibilité d'une scène perçue par un observateur en mouvement ;

*l'utilisation du système : exécution, intégration d'une application.

II. DESCRIPTION DU LOGICIEL PADL-2

II.1 - Structure générale*

Sur le schéma-bloc de PADL-2 (voir figure 2.1) nous trouvons en entrée l'utilisateur dialoguant avec le système. L'ensemble des instructions (définitions et commandes) proviennent soit du clavier alphanumérique (texte frappé par l'opérateur) soit d'un fichier <generic-name> . PFI créé lors d'une session précédente par la commande SAVE (ou sous l'éditeur du système VMS) et appelé par la commande USE <generic-name>.

En sortie nous pouvons obtenir différentes présentations graphiques de l'objet en cours de description: un dessin avec ou sans élimination des parties cachées, une image avec ombrages,... L'interface entre l'utilisateur et la représentation interne des objets est donc, à la fois, textuel et graphique.

Le sous-système ① est le coeur du système de modélisation géométrique: il permet d'entrer, de modifier et de sauvegarder la représentation des objets. L'objet est ici représenté sous forme de GRAPHE. Nous distinguons ensuite un ensemble de processeurs d'application (sous-système ②) qui travaillent à partir de l'ARBRE CSG (arbre binaire représenté sous une forme postfixée), lui-même évalué à partir du graphe. Le premier module, ("SHADER"), permet de produire une image ou un dessin avec élimination des parties cachées par la méthode du Lancer de Rayon ("Ray Casting"). Le second module appelé "Evaluation des frontières" ("BEWIRE") transforme une représentation CSG en représentation par frontières et permet d'afficher un dessin sans élimination des parties cachées (mode "fil de fer"). Le dernier module évalue une représentation par Octrees ("OCTREE") à partir de l'arbre CSG, qui permet de calculer diverses propriétés physiques de l'objet et de le visualiser dans ce nouveau mode de représentation.

---

*Pour mieux comprendre la structure du système, le lecteur devra se familiariser avec le langage et ses fonctionnalités (voir le Manuel Utilisateur de PADL-2)

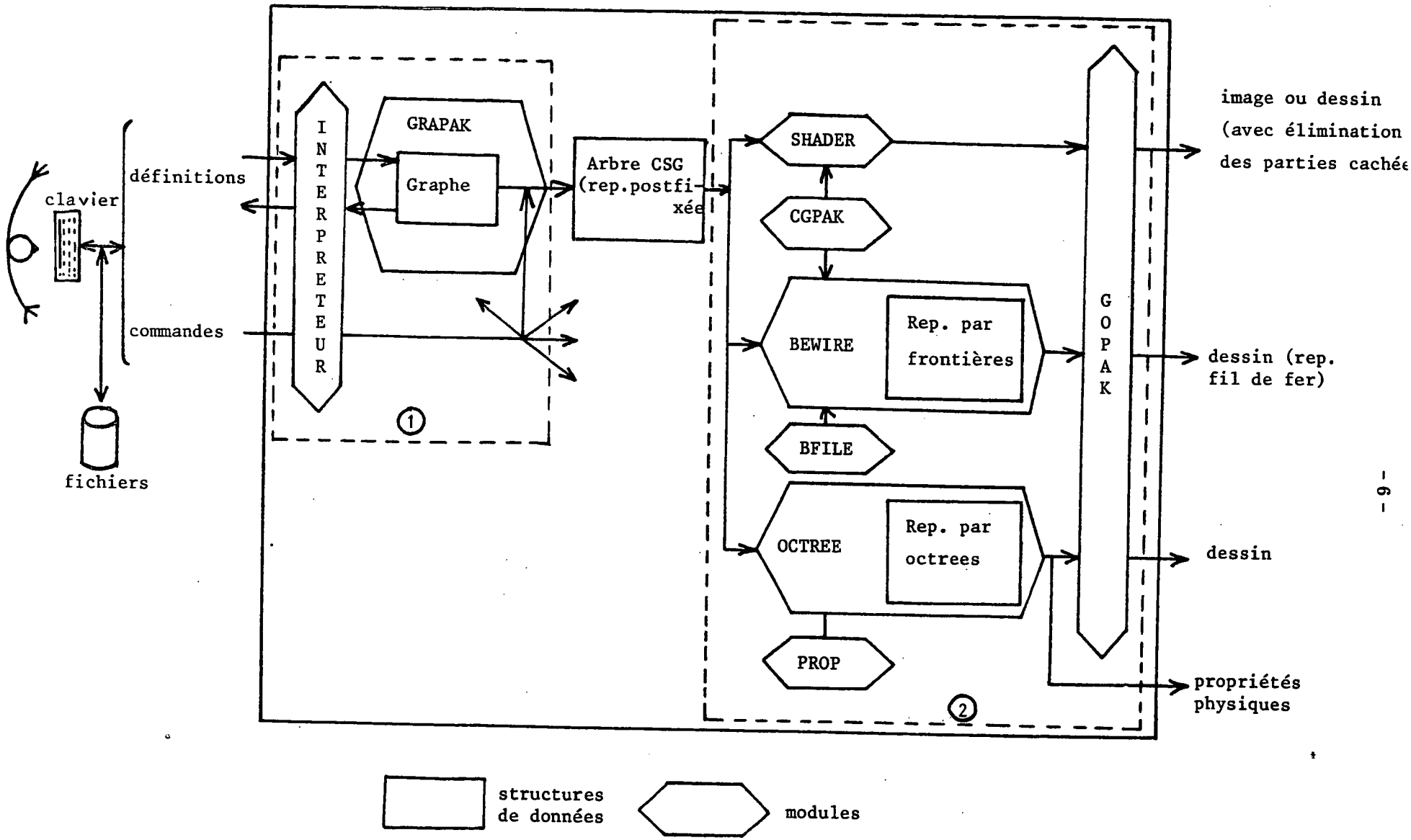


Figure 2.1: Schéma Bloc de PADL-2

II.2 - Représentation CSG

Lors d'une session, les instructions de définition analysées par l'Interpréteur ne sont pas sauvegardées dans leur format initial (chaine de caractères) mais sont transformées sous la forme d'un **Graphe CSG.** Elles sont reconstruites par une transformation inverse du graphe (voir commandes "SHOW" ou "SAVE"). Le graphe est mis à jour au fur et à mesure de l'interprétation des instructions. Les noeuds du graphe sont de quatre types:

- **solide :** invocation d'un nom générique ou d'une primitive géométrique. Combinaison géométrique de solides. Mouvement appliqué à un solide.
- **Système de coordonnées:** mouvement appliqué au repère absolu (LAB), référence à une primitive de type système de coordonnées.
- **nombre réel:** Chaîne de caractères représentant le réel. Expression contenant des opérations réelles monoadiques ou diadiques.
- **liste de paires nom-valeur:** utilisée pour l'invocation des génériques, la spécification des mouvements, l'affectation de valeurs à une liste de paramètres.

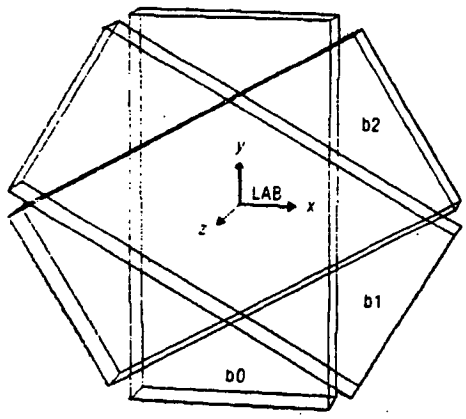
Les noeuds du graphe peuvent être partagés, la seule condition étant que le graphe ne possède pas de cycle. D'autre part, chaque noeud est représenté par son expression littérale et n'est pas évalué (voir figure 2.2).

Pour pouvoir efficacement supporter les calculs mathématiques, on déduit du graphe un **Arbre CSG** où toutes les expressions sont évaluées (expressions réelles, composition de mouvements,...). Les noeuds de l'arbre sont les opérations géométriques (intersection, union,...) et les feuilles sont les primitives géométriques (sphères, cônes,...) décrites par leurs paramètres (hauteur, rayon,...), et leurs positions absolues dans l'espace. L'arbre est décrit dans un tableau sous forme postfixée (voir figure 2.2).

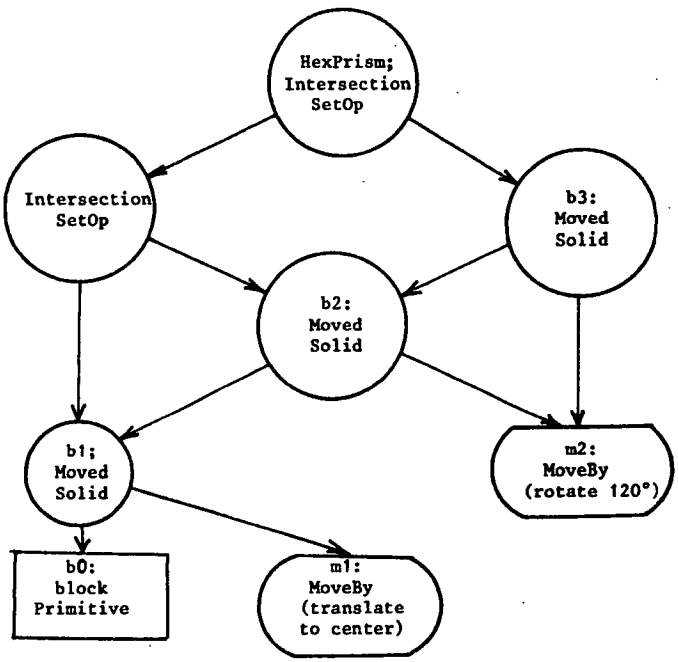
L'ensemble des programmes qui gèrent la représentation CSG de l'objet appartiennent au module GRAPAK (voir documents: SGM-41 "CSG data structure management" et SGM-42 "The PADL-2 post-fix string").

```
GENERIC Hex (HexPrism);
  b0      = BLO(z = Size/4, x = Size, y = 2*Size);
  m1      = (MOVX = -Size/2, MOVY= -Size);
  b1      = MOVE b0 BY (m1) WRT LAB;
  m2      = (ROTZ = (2*3,1416)/3);
  b2      = MOVE b1 BY (m2);
  b3      = MOVE b2 BY (m2);
HexPrism = b1 INT b2 INT b3;
  Size    = 10;
  END;
```

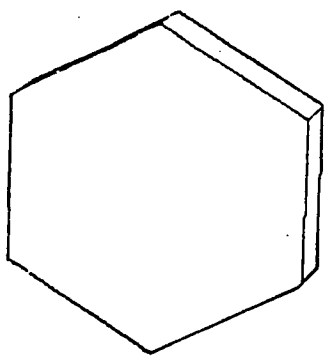
Représentation textuelle de HEX



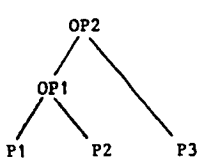
HEX est l'intersection de 3 blocs identiques dans 3 positions différentes



Le GRAPHE CSG de HEX



Dessin de HEX



P1=b0 movedby m1
P2=b0 movedby m2m1
P3=b0 movedby m2m2m1
OPi=Intersection

ARBRE CSG

PFSIndex	1	2	3	4	5
Prims and Ops	P1	P2	OP1	P3	OP2
Motion	m1	m2m1	1	m2m2m1	1
CSGNode	b0	b0	<ptr>	b0	HexPrism
StartLSubTree	0	0	1	0	1
RootLSubtree	0	0	1	0	3
Ancestor	3	3	5	5	0
CongruentTo	2	4	3	1	5

Représentation postfixée de l'arbre CSG

Figure 2.2: Représentations CSG

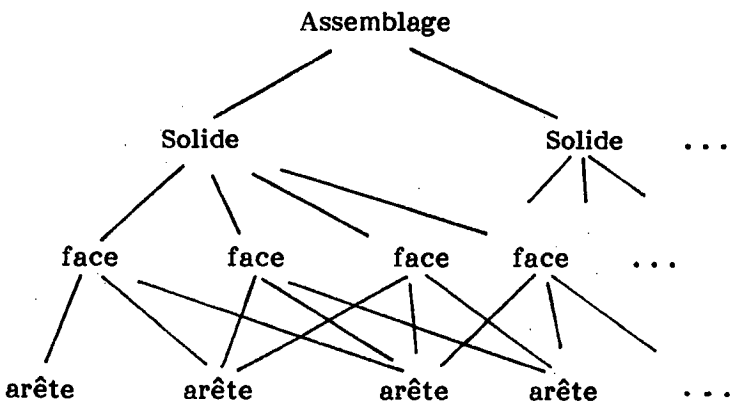
II.3 - Représentation par Frontières

Cette représentation, évaluée à partir de l'Arbre CSG est une structure hiérarchique à quatre niveaux assemblage, solides, faces et contours:

. un solide est un assemblage de faces;

. une face est une surface délimitée par un ensemble d'arêtes (contour)

. une arête est un segment de courbe.



Le passage de l'arbre à la représentation par frontières s'effectue en deux temps:

. une représentation par frontières est évaluée pour chaque feuilles (primitives);

. en chaque noeud, la représentation des frontières est dérivée de celles des sous-arbres gauches et droits.

Cette transformation induit dans la structure la notion de partage des noeuds:

. partage topologique: les faces et les arêtes sont utilisés plus d'une fois;

. partage géométrique: les surfaces, les courbes et les mouvements (positions absolues) sont utilisées plus d'une fois.

Il n'existe cependant pas de partage entre les sous-arbres droits et gauches (voir figure 2.3).

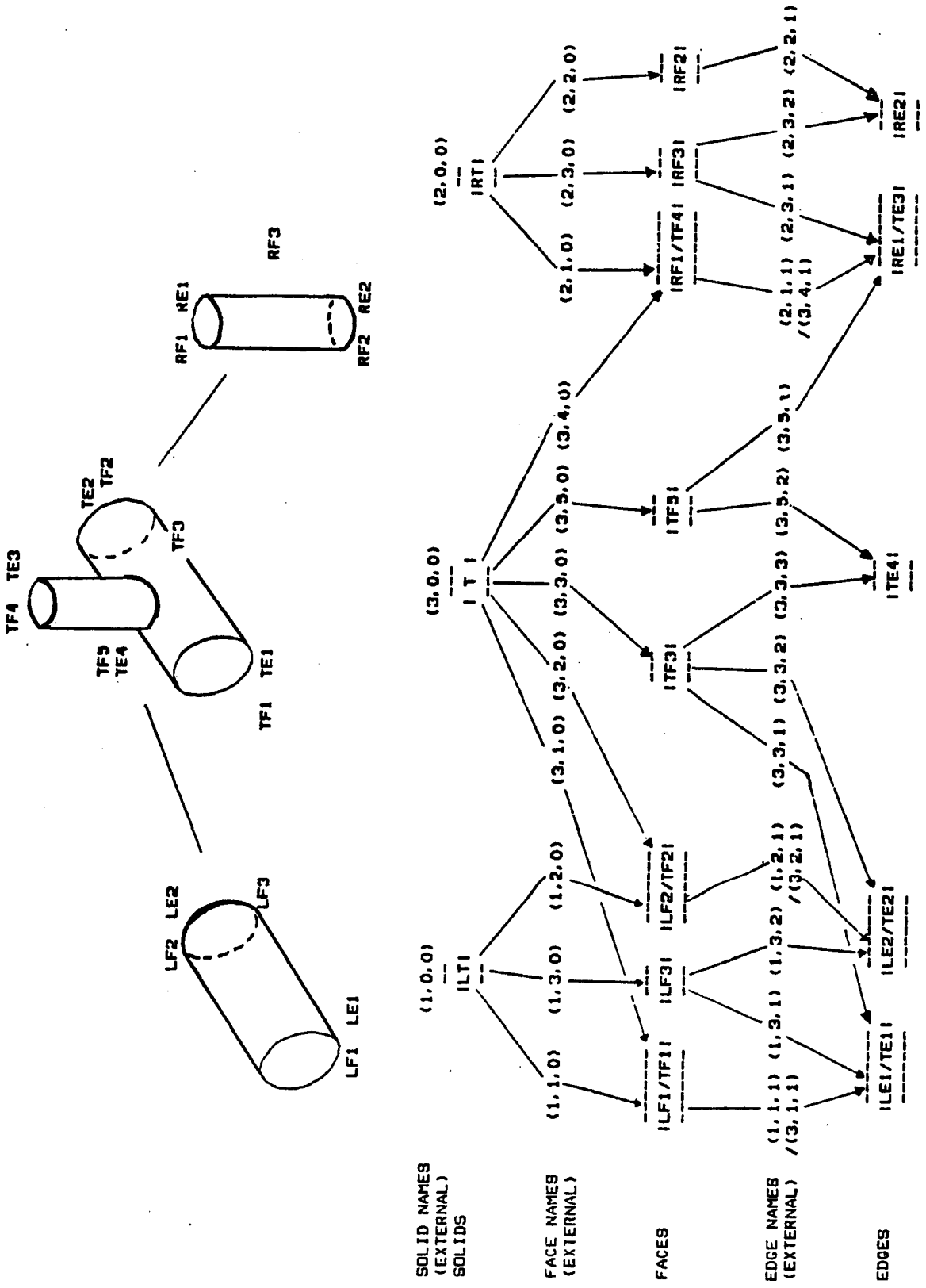


Figure 2.3: Partage topologique

Les entités de cette structure ont trois noms différents:

. les noms donnés par l'utilisateur aux solides et aux assemblages qui sont les chaînes de caractères désignant les noeuds du graphe CSG correspondants;

. les pointeurs donnant l'adresse des entités dans la structure de données;

. les noms externes qui sont des quintuplets d'entiers qui codent la place de l'entité dans la structure: (2, 1, 3, 2, -) désigne par exemple la deuxième arête de la troisième face du premier solide du second assemblage (voir figure (2.3).

Pour évaluer la représentation par frontières, le système manipule quatre types d'**entités géométriques simples:**

. les **primitives solides:** parallélépipède, coin, cylindre, sphère, cône, axe;

. les **demi-espaces:** plan, cylindrique, sphérique, conique et torique. Les primitives solides sont représentées par l'intersection de demi-espaces qui sont les véritables primitives solides manipulées dans PADL-2;

. les **faces** (des primitives solides): rectangle, triangle droit, disque, face cylindrique, face sphérique, face conique et torique;

. les **segments de courbe:** on distingue les segments de courbe simples comme le segment de droite, les arcs d'ellipse, de parabole et d'hyperbole, des segments de courbe qui résultent de l'intersection des diverses faces entre elles comme les combinaisons cylindre/cyclindre, sphère/cylindre, cone/cylindre, cone/sphère, cone/cone.

Les entités géométriques simples sont représentées par leurs équations paramétriques (liste des règles, des paramètres et des contraintes) et par leur position dans l'espace (matrice de transformation linéaire en coordonnées homogènes: rotation + translation). La figure 2.4 est un exemple de représentation du segment de courbe qui résulte de l'intersection d'un cylindre et d'une sphère dans le système de coordonnées local (l'ensemble de ces primitives géométriques sont présentées dans le document CGGM-12 "Representations in the PADL-2 processor: low level geometric entities).

La représentation par frontière est créée et gérée à l'aide des modules BEWIRE et BFILE (voir le document CGGM-20 "BFILE/2: A boundary file for PADL-2").

SPHERE/CYLINDER EDGE SEGMENT

TYPE: "SCEDGE"
CONFIGURATION PARAMETERS: Rc, Rs, h, sign, t0, t1
VALIDITY CONSTRAINTS:

1) Rc, Rs > 0.
2) h > 0.
3) N < D, where N = h**2 + Rc**2 - Rs**2, and
   D = 2*Rc*h.
4) sign = 1 or -1.
5) if N/D < -1, then tmin = $-\pi$, tmax = π .
   if N/D >= -1, then tmax = arccos(N/D), tmin = -tmax. <4>
   tmin <= t0 < t1 <= tmax.

POINT SET: { (x,y,z) : x**2 + y**2 = Rc**2, and
(x-h)**2 + y**2 + z**2 = Rs**2, and
sign*z >= 0 } =

{ (x,y,z) : x = Rc*cos(t),
y = Rc*sin(t),
z = sign*sqrt(rad(t)),
tmin <= t0 <= t <= t0 <= tmax, where
rad(t) = -N + D*cos(t) }.

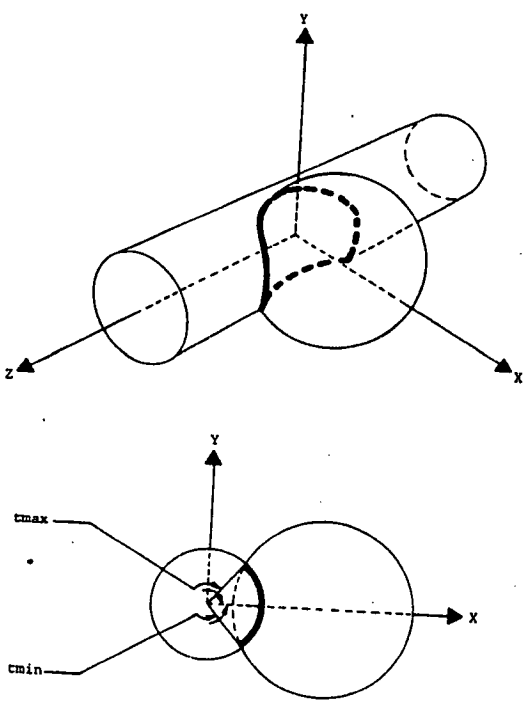
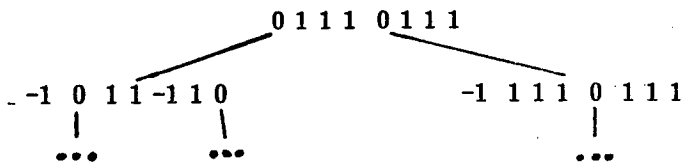


Figure 2.4: Segment sphère/cylindre

II.4 - Représentation par Octrees

La représentation par Octrees est une approximation spatiale de l'objet réalisée à partir de l'arbre CSG. On opère un découpage récursif du cube contenant l'objet en 8 octants. Si l'octant est à l'intérieur ou à l'extérieur de l'objet, ou si on a atteint le nombre de subdivisions maximal on arête le découpage, sinon on continue (l'octant n'est pas entièrement à l'intérieur ou à l 'extérieur de l'objet). La structure de données est simple (voir figure 2.5a) et permet le calcul des propriétés physiques de l'objet (volume, centre d'inertie, moment d'inertie,...).

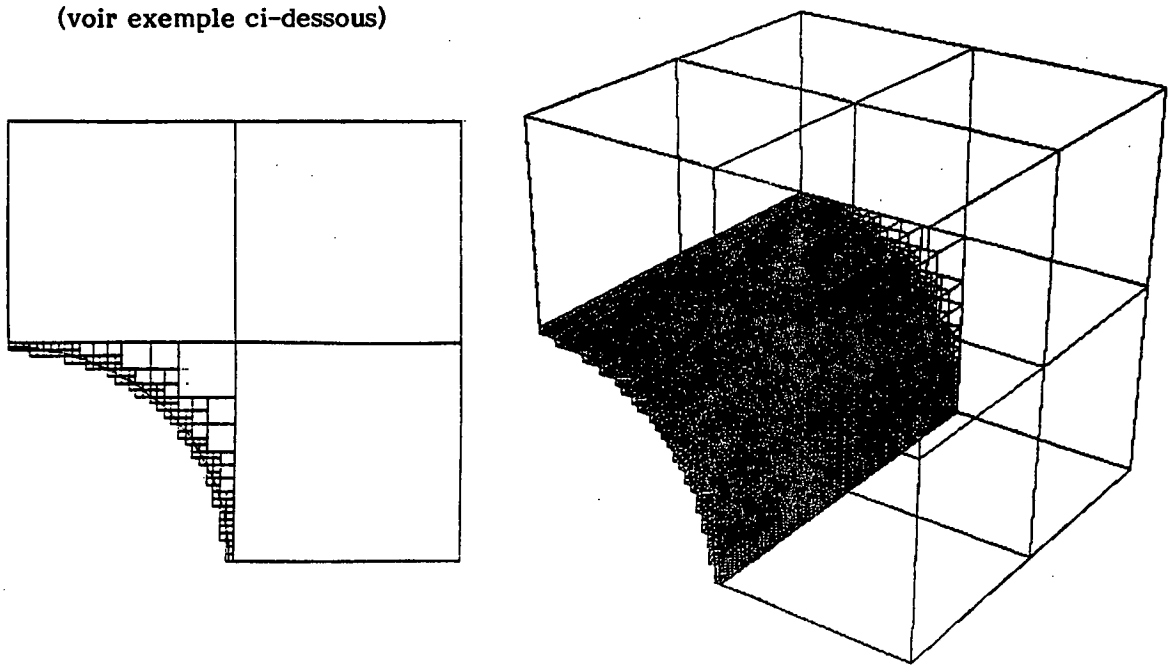


1 = intérieur - 1 = extérieur

0 = pas entièrement à l'intérieur ou à l'extérieur

Figure 2.4: Représentation par Octrees

Lors de l'affichage, seules les cellules marquées -1 et +1 sont visualisées (voir exemple ci-dessous)



Exemple de visualisation d'Octrees

II.5 - Les principaux modules de PADL-2

1. Le module de calcul géométrique: CGPAK

CGPAK est le module de calculs géométriques utilisé pour l'évaluation des frontières et les algorithmes de visualisation.CGPAK permet:

. la création et la gestion des entités géométriques simples

. de réaliser divers calculs et opérations géométriques sur les entités géométriques simples ou de plus bas niveau.

CGPAK fournit des programmes pour:

. opérations d'intersection (entre 2 faces, 2 demi-espaces)

. opérations de classification (classification d'un segment de courbe par rapport à un demi-espace, par rapport à une primitive solide ou la combinaison de deux primitives):

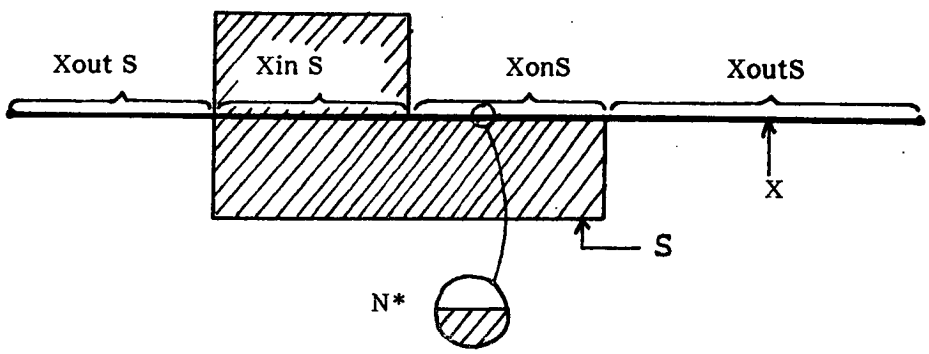
Soient S l'ensemble de référence appartenant à un sous-espace W de E3, et X l'ensemble candidat appartenant à un sous-espace de W.

Les opérations de classification nous permettent de calculer les parties de X qui sont dans S (X in S), sur S (X on S) et hors de S (X out S), et d'extraire les positions relatives des faces de S au voisinage des arêtes X on S (N*(X on S, S)). On note :

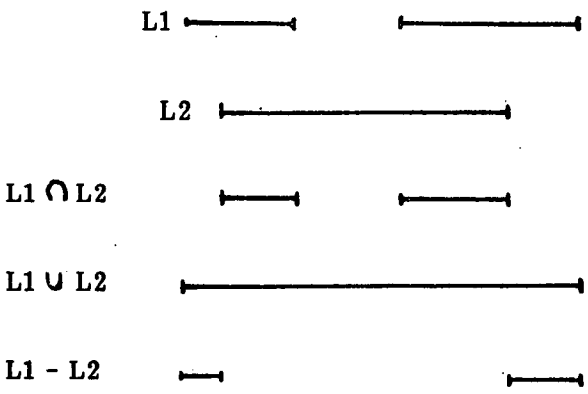
$$M*[X,S] = (XinS, (XonS, N*(XonS,S)), XoutS)$$

avec M pour "Membership classification fonction"

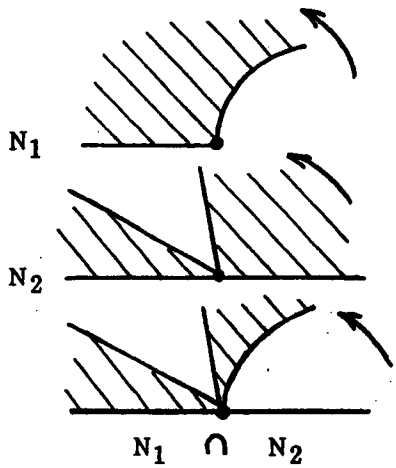
N pour "Neighbourhood"



Lorsque nous effectuons une opération géométrique entre deux solides S1 et S2, nous pouvons également combiner pour un même ensemble candidat X les éléments de M*[X, S1] et de M*[X,S2]



Segments de courbes



voisinage

. opérations sur les segments de courbe (calcul de tangentes)

. opérations sur les surfaces (normale en un point)

. opérations sur les vecteurs (norme, produits scalaires et vectoriels, angle entre deux vecteurs, addition et soustraction de deux vecteurs, multiplication d'un vecteur par un scalaire, normalisation d'un vecteur)

. opérations diverses (création de séquences de points le long d'une courbe, calcul du profil des faces des objets pour un point d'obversation donné, application d'une transformation géométrique sur un point (via RMPAK)).

Les différentes fonctions de CGPAK sont décrites dans le document CGGM-17 "CGPAK: A computational geometry package".

2. Le module de visualisation

Le système graphique de visualisation (voir document IOG-10 "A graphic output processor package") se décompose en trois parties:

PGPAK("Padl-2 Graphic routines): ce sont des sous-programmes d'affichage de haut niveau qui permettent la visualisation des objets, des primitives solides, des repères, des segments de courbe. Nous y trouvons également l'appel au SHADER (voir document IOG-14 "Programmers guide to the PADL-2 shaded graphics output system") qui permet d'obtenir soit une image avec ombrage ou un dessin avec élimination des parties cachées de l'objet par la méthode du lancer de rayon ("Ray Casting"), à partir de sa représentation postfixée.

SGPAK ("Simple graphics package"): c'est le logiciel de préparation à la visualisation. Il permet de définir le type de projection désiré et les paramètres d'observation (point d'observation, point visé, taille de la fenêtre utilisateur,...). Il effectue les projections (transformation 3D -> 2D). On y trouve les primitives d'affichage des segments de droite, des polygones et des boites englobantes.

VGPAK ("Virtual graphics package"): c'est l'interface entre PADL-2 et le terminal de visualisation. On y définit la table des couleurs, les fenêtres écran, l'accès aux pixels. Il permet de spécifier les caractéristiques des terminaux (surfaces pour le dessin au trait ou point par point). Les extrémités des segments de droite sont adressées dans l'espace virtuel normalisé (0,0)x(1,1).

3. Les utilitaires

Nous trouvons des utilitaires qui permettent la création, la gestion, la manipulation et la destruction de:

. piles (STKPAK)

. boites (SPAPAK)

. chaines de caractères (STGPAK) (voir document CRDM-21 "Characters and strings: a policy for character handling in transportable FORTRAN, and a suite of subroutines for creating and manipulating strings")

. mouvements (RMPAK): matrices de transformation 4x4 représentant une rotation plus une translation (voir document CGGM-8 "RMPAK: a set of Fortran subroutines for manipulating rigid motions").

. structures de données (PACPAK): la structure de donnée utilisée est de la forme

(champ 1,...,champ N: élément 1,..., élément i,...)

où le nombre de champs, de types différents, est fixe
la liste des éléments, de même type, est de longueur variable
(voir document CRDM-19 "PACPAK: a set of Fortran subroutines for creating and manipulating data structures").

Un autre module (COMPAK), compare des nombres réels, des vecteurs et des mouvements (matrices 4x4) (voir document CGGM-13 "GMPAK: a collection of Fortran subroutines for computing real numbers, vectors and rigid motions").

Les programmes sources de PADL-2 sont écrits en langage FLECS/77 qui est un surensemble structuré du FORTRAN/77 et qui permet de programmer de façon plus lisible et plus concise que le Fortran standard. Lors de la compilation, un préprocesseur traduit le programme écrit en FLECS en FORTRAN et crée également une version indentée du programme FLECS. Puis la dernière étape consiste à compiler la version en FORTRAN du programme avec n'importe quel compilateur FORTRAN-77 (voir document CRDM-22 "Fleus/77 Users manual").

Il existe enfin un module permettant la gestion de la mémoire en autorisant l'utilisateur à créer et mettre à jour des blocs de mémoire de tailles variées et de changer la taille d'un bloc donné (voir document CRDM-20 "A programmer's guide to SMPAK: a package of Fortran subroutines for stokage management"). SMPAK est utilisé par PACPAK et BFILE 2.

II.6 - Conclusion

PADL-2 a l'avantage d'être un système très ouvert destiné à évoluer et particulièrement bien adapté à la recherche. L'évolution du système peut être à la fois interne (modification du langage, extension des primitives géométriques, accès aux différentes structures de données, etc...) et externe (ajout d'un niveau de description d'un univers 3D, intégration de PADL-2 dans une application,...).

III. EVOLUTIONS DU SYSTEME

III.1 - Modification du langage

L'analyseur syntaxique de l'interpréteur (voir document SGM-40 "The PADL-2 language interpreter") (module PARSER) accepte une grammaire de type LR(1). L'analyse du texte est ascendante et se fait de gauche à droite (séquence inverse de la séquence de dérivation droite).

Possédant le générateur de la table de l'analyseur syntaxique nous pouvons envisager de modifier la grammaire de la manière qui suit:

1 - décrire la grammaire de type LR(1) sous la forme BNF dans le fichier P2-BNF;

2 - lancer la génération de la table de l'interpréteur (exécuter MAKPARSE1.COM),

3 - mettre à jour à la main le nombre de symboles terminaux, non terminaux et le nombre de productions;

4 - Compiler l'interpréteur (exécuter MAKPARSE2.COM);

5 - écrire les instructions et les programmes nécessaires à l'analyse sémantique (procédure SYNTH.FLX et SYNTH2.FLX).

Sans modifier la grammaire nous avons la possibilité d'ajouter des fonctions mathématiques ou des commandes en spécifiant leurs noms et en écrivant les actions sémantiques correspondantes (et les programmes nécessaires) dans la procédure SYNTH.FLX (et dans SYNTH2.FLX pour les nouvelles fonctions mathématiques).

III.2 - Extensions diverses

L'ajout de **nouvelles primitives** comme le tore, l'ellipsoïde, peut être envisagé. Mais celui-ci entraine l'apparition de nouvelles entités géométriques (demi-espaces, surfaces, courbes), d'un nouvel ensemble de calculs géométriques sur celles-ci (intersections, tangentes, normales, opérations de classification, calcul de profil,...) et la modification des diverses structures de données. De manière générale, l'installation d'une nouvelle primitive implique une connaissance parfaite de la totalité du système parce qu'elle engendre des modifications de l'interpréteur aux algorithmes de visualisation, en passant par les évaluations des différents modèles de représentation des objets solides.

La version actuelle de PADL-2 permet de travailler sur différents types de terminaux graphiques:

- Lexidata 3400 (faire SET DEVICE = 1)
- Tektronix 40XX(faire SET DEVICE = 2)
- Gigi (faire SET DEVICE = 6)

Pour installer un **nouveau terminal graphique** nous devons mettre à jour le module VGPAK qui sert d'interface entre le système graphique de PADL-2 et le terminal. On y spécifie les caractéristiques et les nouvelles primitives d'affichage et de gestion de l'écran propres au terminal.

Seule la représentation littérale (texte) des objets est sauvegardée. Pour éviter de refaire à chaque fois toute la chaîne des transformations d'un mode de représentation dans un autre il serait intéressant de pouvoir **sauvegarder et restaurer les différentes structures de données** intermédiaires, en particulier la représentation postfixée de l'Arbre CSG (voir chapitre IV.1).

De nombreux processeurs graphiques permettent maintenant, la visualisation presqu'en temps réel d'objets 3D représentés par des facettes polygonales planes, avec des effets d'ombrages d'ombres portées et de transparences. Le développement d'un algorithme de conversion de l'arbre CSG en **la représentation** del'objet **par un ensemble de facettes polygonales** serait utile pour les applications qui nécessitent de telles performances.

III.3 - Intégration de PADL-2 dans une application

Dans l'état actuel du système il est possible de communiquer avec PADL-2 à partir d'une application via l'interface "Procedural front end" développé à Rochester. Cet interface permet d'invoquer les procédures de PADL-2 à partir d'un sous-programme et résout les problèmes d'appels récursifs à l'interpréteur (voir chapitre suivant).

Cet interface donne l'accès à toutes les ressources de l'interpréteur, des différents modules de création, de gestion et de calcul des diverses structures de données, et des utilitaires. L'inconvénient majeur réside dans le fait que pour tout nouvel objet manipulé ou toute nouvelle occurence de scène nous passons par toute la chaine de transformation: interprétation du fichier littéral, construction du graphe CSG, évaluation de l'arbre CSG, etc... Nous avons développé à cet effet les programmes qui sauvegardent et restaurent l'arbre CSG des objets (voir chapitre IV.1). Un ensemble de sous-programmes seront également développés pour gérer l'arbre CSG (création de l'arbre CSG à partir d'un ensemble d'objets, ajout d'un objet, suppression d'un objet,...).

III.4 - L'interface "Procedural Front End"

Comme nous l'avons mentionné précédemment, cet interface permet d'appeler les procédures de PADL-2 et de résoudre les appels récursifs à l'interpréteur. Au niveau de l'utilisateur, cet interface se compose de deux programmes: **P2INIT** et **PP2** (<instring>, <outstring>, <outvar>).

1. CALL P2INIT ; (initialisations)

Le sous-programme P2INIT doit être appelé exactement une fois avant toute référence à PP2 ou à tout autre sous-programme de PADL-2 ou module. Tous les systèmes de PADL (STG PAK, RM PAK, etc...) sont initialisés par P2INIT.

2. CALL PP2 (<instring>, <outstring> , <outvar>); (communications)

Le sous-programme PP2 met en oeuvre les communications avec PADL-2.

La variable <instring> est une chaine de caractères FORTRAN-77 contenant n'importe quelle instruction du langage de PADL-2 pour être envoyée à l'interpréteur de PADL-2. <instring> peut avoir une longueur de 130 caractères. Cependant il est à noter que pour les sorties graphiques et l'accès aux bibliothèques par des noms génériques, certains noms logiques doivent être valides dans l'environnement VAX/VMS. Suggestion: exécutez votre programme en utilisant une version du fichier de commandes qui invoque les possibilités interactives du système. La chaîne <insting> peut être une constante, la procédure PP2 ne la modifiant pas.

La variable <outstring> devant être déclarée CHARACTER * 132 en FORTRAN-77, permet le cas échéant de recevoir des informations de PADL-2.

La variable <outvar> devant être déclarée REAL*4 en FORTRAN-77, permet de recevoir des informations numériques en provenance de PADL-2. Dans la version actuelle aucune valeur n'est renvoyée par le système.

Ces deux procédures sont destinées à remplacer tous les appels à l'ensemble courant des programmes (GRAxxx) de GRAPAK. Tant que les fonctions de sortie ne sont pas encore toutes écrites, les utilisateurs trouveront certainement utile la fonction entière GRAGET. Par exemple, pour déterminer la position d'un repère (définit comme un mouvement rigide - "Rigid Motion" -) appelé SLOT faire :

```
INTEGER ISLOT, IRM, GRAGET
INCLUDE "P2GRA : GRACOM . CMN"
REAL U(3), V(3), W(3), P(3)
ISLOT = 0
CALL INIT (ISLOT, "SLOT", 4)
IRM = GRAGET (ISLOT, GRARM)
CALL GETRM (IRM, U, V, W, P)
CALL KILLRM (IRM)
CALL RLSE (ISLOT)
```

la position désirée est dans P(1), P(2), P(3).

IV. EXTENSIONS REALISEES

IV.1 - Nouvelles commandes

De nouvelles commandes sont disponibles sous l'interpréteur de PADL-2:

. WBFL [<nom-objet>]

Edition lisible de la représentation par frontière (BFILE) de l'objet <nom-objet> (par défaut <nom-de-la-racine> de l'arbre CSG) sur l'écran.

. WCSG [<nom-objet>]

Sauvegarde de la chaine postfixée représentant l'arbre CSG de l'objet <nom-objet> (par défaut <nom-de-la-racine> de l'arbre CSG) dans le fichier <nom-objet> • CSG (par défaut dans le fichier <nom-de-la-racine> • CSG).

. DCSG <nom-objet>

Visualisation fil de fer de l'objet <nom-objet> à partir de sa représentation par frontière (BFILE) évaluée après la lecture du fichier <nom-objet> • CSG. Seul l'arbre CSG existe mais pas le graphe. Aussi aucune autre fonction ne peut-être pour le moment, appliquée à l'objet.

. CREER <nom-univers>

Appel à l'interpréteur du logiciel de description d'une scène 3D (tridimensionnelle). Nous décrivons ce logiciel dans le paragraphe suivant.

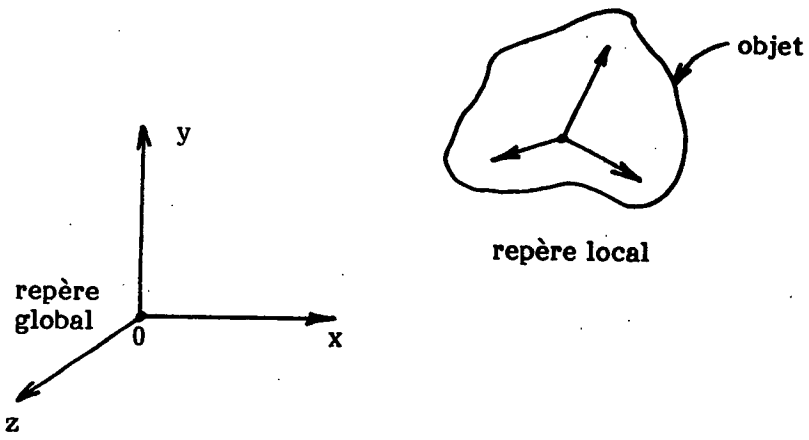
IV.2 - Logiciel de description d'un Univers 3D

a. Généralités

La version de ce logiciel décrit ici est élémentaire dans le sens où:

. chaque objet possède soit un seul niveau (l'objet est une pièce indivisible) soit deux niveaux (l'objet est un assemblage de pièces).

. chaque objet auquel est associé implicitement un repère local connu de l'utilisateur, est positionné de façon absolue par rapport au repère univers (global)



Si nous restons dans le contexte de PADL-2, cet outil de description de scènes 3D n'apporte rien par rapport aux possibilités offertes par le langage de PADL-2. En effet ce dernier permet de créer facilement un univers 3D par un assemblage d'objets positionnés de manière absolue ou relative (les objets pouvant être invoqués par leur nom générique et paramètrés).

L'intérêt majeur de ce logiciel est de décrire un univers 3D indépendamment du modèle de représentation des objets. Ainsi pour une même scène plusieurs bases de données peuvent être créées, une pour chaque type de représentation. Dans ce qui suit les différents modèles décrivant les objets sont obtenus à partir de PADL-2, mais ils pourraient provenir d'un autre système de modélisation.

b. Le langage

1 - Description

Un **objet** est décrit dans le repère absolu de l'univers selon la syntaxe suivante :

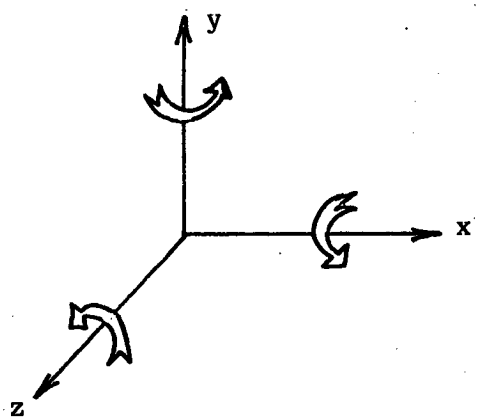
<objet> :: = - <nom-objet> [: <position>]

<position> :: = <mouvement-élémentaire> [, <mouvement-élémentaire]*

<mouvement-élémentaire> :: = <nom-mouvement> = <cste-réelle>

<nom-mouvement> :: = MOVX|MOVY|MOVZ (translations)
|DEGX|DEGY|DEGZ (rotations en degrés)
|ROTX|ROTY|ROTZ (rotations en radians)

le sens positif de chaque rotation est décrit sur la figure ci-dessous



exemple :

- US : DEGX = -90, MOVZ = 50, MOVX = 40

2 - Commandes de mise à jour

Les trois commandes de mise à jour ont pour syntaxe:

<commande> :: = AJOUTER|A (ajout)
|MODIFIER|M (modification)
|SUPPRIMER|S (suppression)

chacune de ces commandes attribue au contexte l'opération qui lui correspond (ajout, suppression, modification).

L'opération de mise à jour est appliquée sur la liste des objets définis ensuite tant qu'une nouvelle commande n'est pas effectuée.

3 - Création des bases de données

Pour un même Univers 3D, quatre fichiers différents peuvent être créés:

. fichier <nom-univers> • PFI où l'Univers est décrit dans le langage de description de PADL-2. Ce fichier permettra de visualiser l'univers décrit (dessin ou image),

. fichier <nom-univers> • PFL où la scène est décrite dans le langage de description de l'Univers 3D. Les diverses scènes 3D pourront être mises à jour ou réutilisées pour la description de nouvelles scènes,

. fichier <nom-univers> • PFS où l'Univers est décrit dans le langage de description du logiciel de Synthèse d'Image (SI) du banc de simulation associant l'image et le mouvement pour la commande de robots (convention IRISA-THOMSON LER),

. fichier <nom-univers> • CSG où l'Univers est représenté par la chaine post-fixée de son arbre CSG, et visualisable sous l'interpréteur de PADL-2 par la commande DCSG <nom-univers>.

Ces quatre fichiers seront créés en tapant les commandes suivantes:

<commande>:: = CPFI (création du fichier <nom-univers> • PFI)
|CPFL (création du fichier <nom-univers> • PFL)
|CPFS (création du fichier <nom-univers> • PFS)
|CCSG (création du fichier <nom-univers> • CSG)

La base de donnée 3D <nom-univers> • PFS, respectivement <nom-univers> • CSG, est constituée à partir des fichiers <nom-objet$_i$> • PFS, respectivement <nom-objet> • CSG, obtenue sous l'interpréteur de PADL-2 pour chaque objet$_i$ appartenant à l'Univers 3D (voir commande WPFS au chapitre IV.3.d, respectivement WCSG au chapitre IV.1).

4 - Commandes diverses

Pour utiliser et insérer une ancienne scène 3D dans le contexte on tape

<commande> :: = UTILISER|U

puis séquentiellement le (ou les) nom de l'Univers 3D désiré

<univers> :: = -<nom-univers>

Les objets contenus dans le fichier <nom-univers> • PFL sont ajoutés au contexte.

Pour obtenir la liste des objets contenus dans le contexte faire

<commande> :: = LISTE|L

Pour sortir de l'interpréteur faire

<commande> :: = FIN

4 - Exemple

Voici un exemple d'Univers 3D de nom "UNIV" présenté sous trois formes:

1. fichier UNIV.PFL
2. fichier UNIV.PFI
3. dessin de UNIV

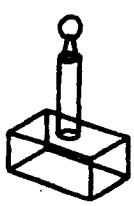
1. UNIV.PFL;

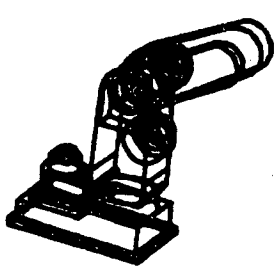
- CL
- US : DEGX = -90 , MOVZ = 50 , MOVX = 40
- SCEMI1 : MOVX = 40 , MOVZ = -20

2. UNIV.PFI;

```
GENERIC UNIV(UNIV)
UO=CL ( )
U1=U0 ASB US( ) MOVEDBY (DEGX = -90 , MOVZ = 50 , MOVX = 40 )
U2=U1 ASB SCEMI1( ) MOVEDBY (MOVX = 40 , MOVZ = -20)
UNIV = U2
```

3. dessin





c. Règles pour la description des objets

Les objets sont décrits un par un sous l'interpréteur de PADL-2 (voir le manuel utilisateur de PADL-2, document UM-10/1.2).

Pour faciliter la tache de l'utilisateur lors de la description et de la manipulation de la base de données 3D <nom-univers> . PFS, les règles de programmation qui suivent devront être respectées:

1 - Déclaration des Génériques

Lors de la description d'un objet solide on attribue un nom au contexte courant par l'instruction :

GENERIC <extname> (<root-solid>)

où <extname> est un nom externe qui procure le nom du fichier correspondant (<extname> . PFI) lors de l'invocation de la commande SAVE,
<root-solid> est le nom interne de n'importe quel solide du contexte.

Afin d'identifier les noms externes et internes de l'objet, l'opérateur déclarera les génériques de la manière qui suit :

GENERIC <nom-objet> (<nom-objet>)

le nom interne et le nom externe (fichier) de la pièce propriétaire seront identiques.

2 - Description des objets

Un objet est défini par un assemblage de pièces.

<objet> = <pièce 1> ASB <pièce 2> ... ASB <pièce N>

où $N \geqslant 1$

Lorsque nous manipulons l'arbre CSG de l'objet, nous identifions les pièces par leurs noms. Il faut savoir que si au cours de la description de l'objet dans le langage de PADL-2 on écrit :

<nom1-pièce$_i$> = <définition pièce$_i$>

<nom2-pièce$_i$> = <nom1-pièce$_i$> AT <coord. syst.>

.
.
.

<nom-objet> = ... ASB <nom2-pièce$_i$> ASB ...

ce n'est pas <nom2-pièce$_i$> que l'on récupère dans l'arbre CSG (chaîne postfixée) pour la pièce$_i$ mais <nom1-pièce$_i$>

3 - Invocation des génériques

Un objet peut être invoqué en utilisant son nom générique (nom du fichier correspondant) suivi par des parenthèses contenant éventuellement une liste de paramètres :

<nom-objet> (<liste de paramètres>)

Si cette instruction autorise l'utilisation d'objets déjà créés, elle ne nous permet pas d'identifier dans l'arbre CSG les objets et les pièces qui les composent par leurs noms respectifs. Lors de la transformation du graphe CSG en arbre CSG les sous-graphes correspondant aux noms génériques sont détruits et en particulier les noms des éléments donnés par l'utilisateur. Donc si nous réutilisons un objet déjà créé qui constituera l'une des pièces de l'objet en cours de création et qui pourrait être lui-même formé d'un assemblage de pièces, nous l'insérerons dans le contexte par la commande USE <nom-objet> en modifiant si nécessaire certains paramètres ou certains noms de variables préexistants dans le contexte. On s'interdit ainsi l'invocation des Génériques.

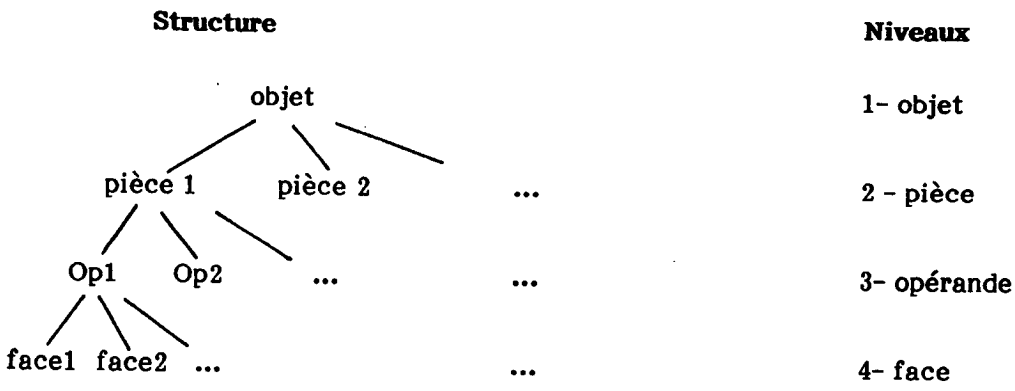
d. Création des fichiers <nom-objet> • PFS

Les fichiers <nom-objet> • PFS décrivent les objets dans le langage de description du logiciel de Synthèse d'Image (SI). Ils sont créés sous l'interpréteur de PADL-2 en tapant la commande

WPFS [<nom-objet>]

pour ce faire, il faut qu'initialement l'objet appartienne au contexte courant.

Toutes les informations nécessaires à la création du fichier <nom-objet>.PFS sont extraites automatiquement de l'arbre CSG à l'exception des modèles de surface. Ces derniers, référencés par leurs noms, sont attribués aux différentes faces des opérandes constituant l'objet (bloc, sphère, cone, cylindre, coin,...) de manière interactive. D'autre part l'attribution d'un modèle de surface se fait suivant une analyse descendante de la structure hiérarchique à 4 niveaux des objets: objet, pièces, opérandes, faces.



Un nom de surface donné à un noeud de l'arbre, de type objet pièce ou opérande, est attribué implicitement aux feuilles (c'est à dire aux faces des opérandes) du sous-arbre correspondant.

Les noms de surface sont choisis dans un catalogue (fichier BD SURF.SUF) où pour chaque surface sont donnés le nom de la surface, le nom du modèle et la liste des divers coefficients correspondants. Ce catalogue peut être mis à jour à tout moment.

Pour chaque fichier <nom-objet>.PFS est constitué automatiquement le fichier <nom-objet>.SUF qui contient la liste des surfaces qui décrivent l'objet. Ces fichiers <nom-objet>.SUF sont utilisés pour générer lors de la création de <nom-univers>.PFS, le fichier <nom-univers>.SUF qui nous procurera la liste des surfaces employées pour décrire l'Univers.

e. Conclusion

Ce logiciel de description d'Univers 3D permet actuellement de fournir pour une même scène 3D quatre fichiers différents (*.PFI, *.PFS, *.PFL, *.CSG). Un autre modèle est envisagé qui décrira la scène dans un langage de description d'un autre logiciel de synthèse d'image par "lancer de rayon" (Equipe de Kadi Bouatouch, IRISA). La technique utilisée sera équivalente à celle employée pour la création des fichiers *.PFS.

La mise en oeuvre de ce système simple de description de scènes 3D nous a permis de répondre à plusieurs objectifs:

. évaluer PADL-2: utilisation des sous-programmes des différents modules et des utilitaires, compréhension et manipulations des diverses structures de données associées aux modèles géométriques,

. fournir rapidement des bases de données 3D pour la mise au point des logiciels de synthèse d'image,

. tester les nouveaux algorithmes utilisés pour manipuler les structures de donné pour réaliser des calculs géométriques sur la scène,...

Un système complet de description d'Univers 3D est en cours d'étude, incluant la coexistence de plusieurs modèles d'objets, diverses structures relationnelles des objets en fonction d'informations d'ordre géométrique, topologique et autres, et la description de mécanismes articulés.

Une fois cet univers 3D décrit, il est nécessaire d'en extraire la partie visible pour l'observateur. Dans le cas de la génération d'images synthétiques statiques, le problème est résolu par un algorithme de "clipping". Cependant si l'on considère la génération de séquences d'images vues par un observateur se déplaçant au sein de l'univers 3D, le problème de la gestion en temps réel des objets apparaissant ou disparaissant du champ de vision se révèle plus complexe. Un algorithme permett sous certaines hypothèses simplificatrices, cette gestion est présentée dans le pr chapitre.

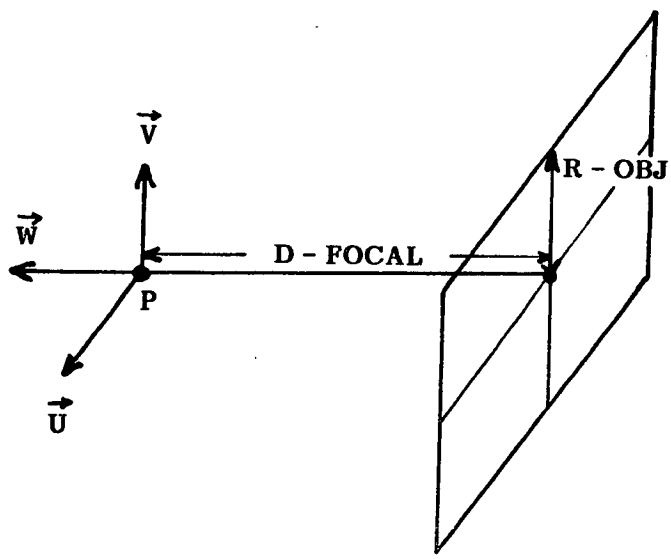
IV.3 - Intégration du mouvement dans PADL-2

De nombreuses applications comme la simulation en robotique ou la production de séquences animées, introduisent la notion de mouvement que ce soit celui de l'observateur ou celui d'objets mobiles évoluant dans la scène. Mais dans PADL-2, aucun mécanisme n'autorise la mise en œuvre du déplacement d'un observateur ou d'un objet autrement que par l'intermédiaire du langage, c'est-à-dire via l'interpréteur. Nous avons donc développé un ensemble de sous-programmes qui modifient la position des objets directement au niveau de leurs représentations, ainsi que les paramètres de visualisation.

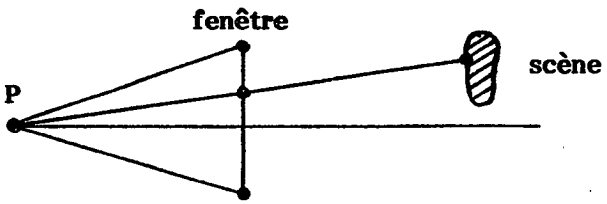
IV.3.1 - Observateur en mouvement

Afin de pouvoir visualiser la séquence des images perçues par un observateur en mouvement un nouveau sous-programme SGCAME (U, V, W, P, D-FOCAL, R-OBJ) permet de mettre à jour à chaque pas les divers paramètres de visualisation. L'observateur est assimilé à une caméra dont les caractéristiques sont spécifiées de la manière suivante :

- position : P	(real P(3))
- orientation : repère (U,V,W)	(real u(3), v(3), w(3))
- distance focale : D-FOCAL	(real D-FOCAL)
- rayon de l'objectif : R-OBJ	(real R-OBJ)



Le rayon de l'objectif définit une fenêtre carrée de côté 2 x R-OBJ. Le vecteur W est situé sur l'axe optique et il représente la direction du point visé (centre de la fenêtre) vers le point d'observation P . Le dessin ou l'image perçue à un instant donné est obtenu par la projection centrale de la scène sur la fenêtre :



Le mouvement de la caméra et la modification de ses paramètres (D-FOCAL par exemple) sont gérés par le programme d'application.

IV.3.2 - Gestion des objets mobiles

La première solution envisageable pour déplacer un objet dans une scène consiste à utiliser le langage de PADL-2 : on modifie l'expression décrivant le solide en y spécifiant sa nouvelle position. Mais cette méthode n'est pas simple dans son utilisation et n'est pas optimale, en particulier dans le cadre d'applications hors ligne.
D'autre part, elle s'avère caduque si nous manipulons la scène directement à partir de son arbre CSG.

La seconde solution que nous avons développée applique le mouvement de l'objet directement sur son arbre CSG et sur sa représentation par frontière. Une séquence typique d'opérations mettant en œuvre cette solution aura la forme qui suit :

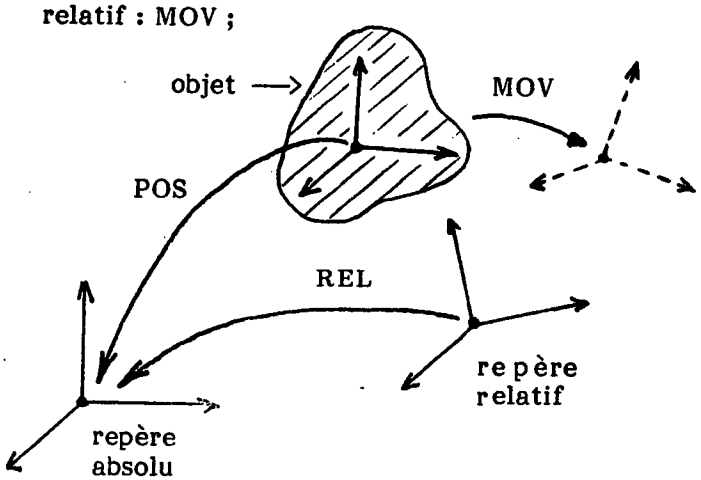
1 - obtenir l'arbre CSG de la scène initiale ;
2 - évaluer la représentation par frontière de la scène ;
3 - afficher la scène initiale (dessin ou image) ;

4 - calculer le mouvement de chaque objet mobile ;
5 - mettre à jour l'arbre CSG et la représentation par frontière des objets mobiles ;
6 - afficher la scène courante ;
7 - répéter 4, 5 et 6 pour chaque occurence de la scène.

Si nous visualisons la scène à l'aide de l'algorithme de lancer de rayon (SHADER) nous utilisons uniquement l'arbre CSG, le calcul et la mise à jour de la représentation par frontière deviennent inutiles.

Les types de mouvements que l'on peut appliquer sur un objet peuvent varier à l'infini en combinant les rotations et les translations élémentaires, et les divers repères dans lesquels on les effectue. Pour chaque mouvement élémentaire on utilise :

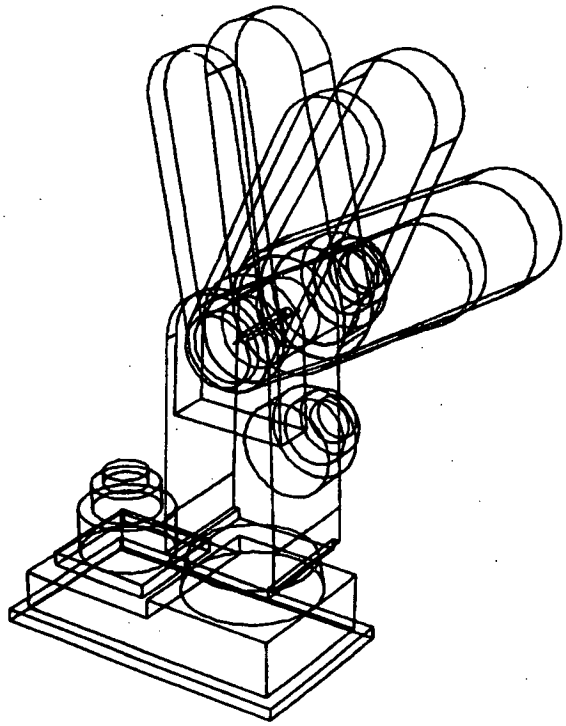
1 - la position du repère local lié à l'objet dans le repère absolu : POS ;
2 - la position absolue du repère relatif par rapport auquel s'effectue le déplacement de l'objet : REL ;
3 - la valeur du mouvement élémentaire appliqué à l'objet dans le repère relatif : MOV ;



La nouvelle position absolue de l'objet après le mouvement est donnée par l'expression :

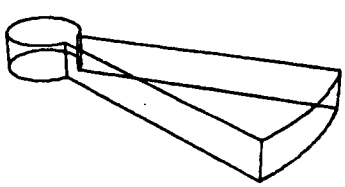
$$REL \circ MOV \circ REL^{-1} \circ POS$$

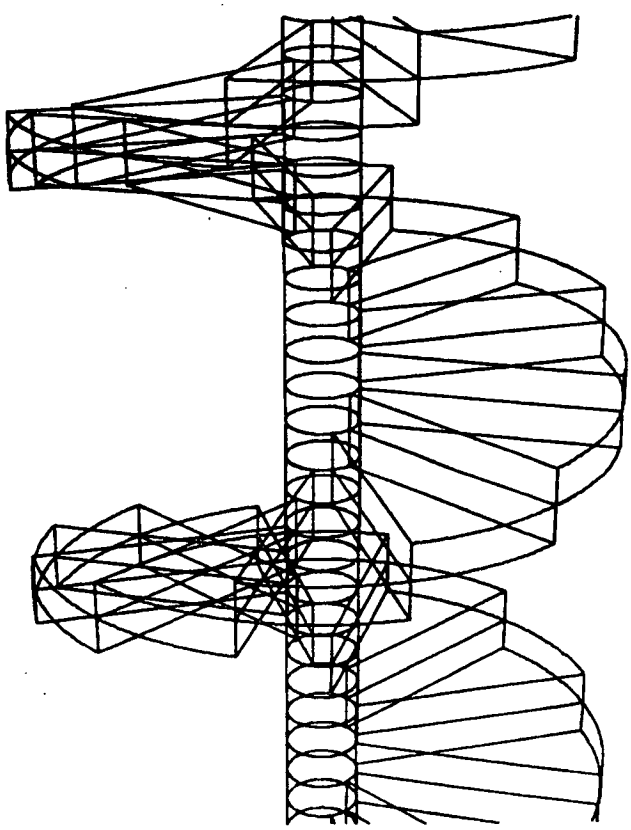
Objets mobiles: exemples d'application



Simulation du mouvement d'un robot.

Visualisation d'une scène (ici un escalier) à partir de la répétition d'une primitive de base (ici une marche) par applications successives d'un mouvement élémentaire.





V. GESTION DYNAMIQUE D'UNE SCENE EN FONCTION DU DEPLACEMENT DE L'OBSERVATION

V.1 - Présentation

Les systèmes de générations d'images synthétiques utilisent des bases de données d'objets 3D de plus en plus complexes pour décrire de façon réaliste une scène.

Dans le cas de la création d'images de synthèse fixes, cette base de données est créée de façon statique mais en ce qui concerne la génération de séquences d'images, domaine qui se développe très rapidement que ce soit en audiovisuel ou dans le cas des simulateurs de pilotage, il est nécessaire de prendre en compte une évolution dynamique de la scène en fonction des mouvements de l'observation et des objets mobiles qu'elle peut contenir.

Souvent, des contraintes temporelles très strictes sont imposées au système de synthèse (Ex.: simulateur de vol) et nécessitent des algorithmes capables de gérer la base de données en temps réel - c'est-à-dire capables à tout instant de déterminer les objets apparaissant ou disparaissant du champ de vision de l'observateur.

Lors de la conception de tels algorithmes, on se heurte, le plus souvent, au problème de la manipulation de structures de données sophistiquées (arbres, graphes, ...) qui représentent des modèles géométriques d'objets très complexes et qui sont mal adaptés à des manipulations Temps Réel.

La méthode proposée associe au modèle complet de l'objet, un modèle beaucoup plus simple constitué par un certain nombre de ses points caractéristiques. Celà dans le but de ramener le problème de la gestion dynamique d'objets 3D complexes à un problème de gestion dynamique d'un ensemble de points reliés entre eux de façon rigide.

Dans une séquence d'images on peut distinguer deux types de mouvements :

- les mouvements dus à la présence d'objets mobiles dans la scène
- les mouvements dus au déplacement de l'observateur (capteur) dans la scène.

C'est ce dernier type de mouvement qui sera examiné ici en tenant compte des hypothèses suivantes :

- Les objets contenus dans la scène sont immobiles et leurs coordonnées sont connues lors de la constitution de la base de données 3D associée à la scène.

- Pour des raisons d'isotropie, on considérera que pour une position donnée de l'observateur dans la scène les seuls objets susceptibles d'être perçus sont ceux contenus dans une boucle de rayon R paramétrable (portée de la "vision") et centrée sur la position de l'observateur.

- La direction du vecteur déplacement Vd de l'observateur (capteur) est connue à chaque instant et son amplitude maximale est bornée par une valeur Vd_{max} connue et inférieure au rayon R de la sphère de vision ($|Vd| \leq Vd_{max} < R$).

La seconde hypothèse se justifie par la volonté de ne pas avoir à recalculer la base de données partielle dans le cas d'une rotation pure de l'observateur ("panoramique").

Moyennant ces hypothèses le problème peut s'énoncer ainsi :

Soit un univers fini $\mathcal{U}$ dans l'espace Euclidien E^3 constitué d'objets ponctuels O_i dont les coordonnées (x_i , y_i , z_i) sont connues dans un repère de référence $R_{\mathcal{U}}$, soit un sous-ensemble V (B.D. partielle) de $\mathcal{U}$ constitué par les éléments de $\mathcal{U}$ contenus dans une boule fermée de rayon R, centrée sur un point C dont les coordonnées (x_C , y_C , z_C) sont connus dans R et se déplaçant dans $\mathcal{U}$ suivant un vecteur $\mathcal{V}_d$. Peut-on gérer dynamiquement l'ensemble des objets présents dans V à l'instant t_n en minimisant le temps de calcul et en le rendant indépendant du nombre total d'objets contenus dans $\mathcal{U}$?

Résoudre ce problème revient à :
- minimiser le nombre d'objets traités entre t_{n-1} et t_n
- minimiser le nombre d'opérations par objet (surtout en flottant)

La première partie sera consacrée à la définition des structures de fichiers associées aux objets de l'univers $\mathcal{U}$.

Dans la seconde partie sera abordé le problème de la modélisation et de la gestion dynamique de la base de données locale constituée par les objets présents dans la sphère de vision à un instant t.

Enfin, la dernière partie présentera les résultats obtenus en simulation et permettra d'évaluer les performances de l'algorithme.

V.2 - Structure informatique associée à la base de données "UNIVERS $\mathcal{U}$ "

La base de données "Univers" comporte deux niveaux de description, un premier niveau constitué par les représentations 3D complètes des objets contenus dans l'univers $\mathcal{U}$ et un second niveau, déduit du premier, qui associe à chaque représentation d'un objet un ensemble de points caractéristiques de celui-ci (sommets, points de la surface, points du squelette, etc.). Dans ce papier nous considérerons que l'association d'un objet complexe à un ensemble de points caractéristiques de sa surface est réalisée et c'est sur cet ensemble de points que portera notre algorithme. A ce deuxième niveau de représentation, utilisé par l'algorithme de gestion dynamique, sont associés deux fichiers :

- un fichier caractérisant les relations géométriques.
- un fichier caractérisant les relations topologiques.

V.2.1 - Relations géométriques

Ce fichier attribue un pointeur qui constituera le numéro absolu de référence du point représenté par ses coordonnées dans le reprère R .
En plus de ses coordonnées, une autre variable est associée à chaque point et utilisée par l'algorithme de gestion dynamique pour spécifier la distance du point vis-à-vis de l'observateur.

La structure de ce fichier est donnée à la figure 1.

TAB 4(NUMOBJ,L)	X(NUMOBJ)	Y(NUMOBJ)	Z(NUMOBJ)	DIST

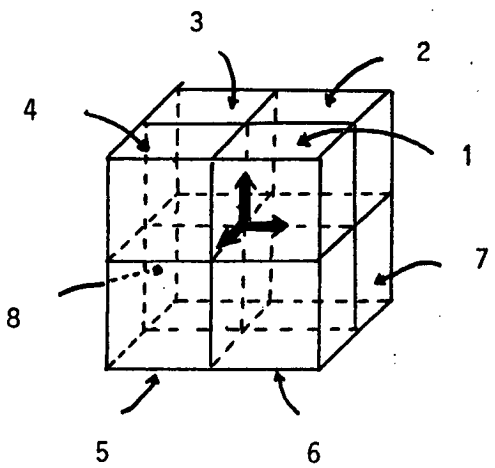
Figure 1 : Structure du fichier TAB 4

Ce fichier regroupe toutes les informations en format flottant nécessaires à l'algorithme de gestion dynamique de la base de données partielle V.

V.2.2 - Relations topologiques

Ces relations caractérisent l'environnement local de chaque point dans l'univers $\mathcal{U}$.

A chaque point i, on associe un repère R_i obtenu par translation du repère de référence R_V ; ce repère R_i permet de partitionner l'espace en huit sous-espaces connexes $\{ \mathcal{E}_1, \ldots, \mathcal{E}_8 \}$ (figure 2).



$x \geq 0$	$y \geq 0$	$z \geq 0$
$x < 0$	$y \geq 0$	$z \geq 0$
$x < 0$	$y < 0$	$z \geq 0$
$x \geq 0$	$y < 0$	$z \geq 0$
$x \geq 0$	$y < 0$	$z < 0$
$x \geq 0$	$y \geq 0$	$z < 0$
$x < 0$	$y \geq 0$	$z < 0$
$x < 0$	$y < 0$	$z < 0$

Figure 2 : Relations topologiques entre objets

Pour chaque point i de l'univers $\mathcal{U}$ on calcule les huits plus proches voisins dans les sous-espaces $\mathcal{E}_1, \ldots, \mathcal{E}_8$ au sens d'une distance euclidienne. Celà permet d'associer aux N points de l'univers $\mathcal{U}$ une structure de graphe orienté préservant les propriétés topologiques de cet univers.

Dans la structure de fichier associé à ce graphe (fig. 3) chacun des plus proches voisins du point considéré est codé par son numéro absolu ; de plus, un certain nombre de pointeurs supplémentaires est associé à chaque point et dont le rôle, qui sera précisé ultérieurement, est d'effectuer les chaînages au sein du graphe lors de la gestion dynamique de la base de données partielle $\mathcal{V}$.

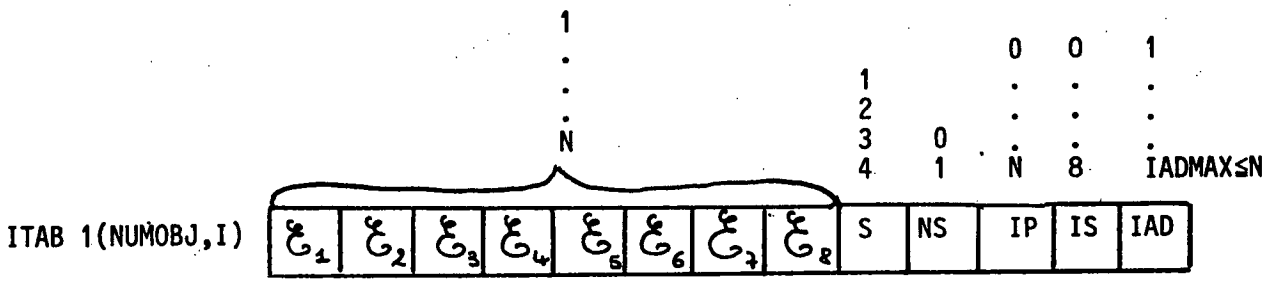


Figure 3 : Structure du fichier ITAB 1

V.3 - Modélisation et gestion de la base de données partielle

En utilisant les hypothèses posées précédemment sur le modèle sphérique de la vision locale et sur le déplacement du capteur, on peut partitionner les objets de l'univers $\mathcal{U}$ à l'instant t_n en quatre classes en fonction de leur possibilité d'appartenir à la base de données partielle $\mathcal{V}$ à l'instant t_{n+1} (fig. 4).

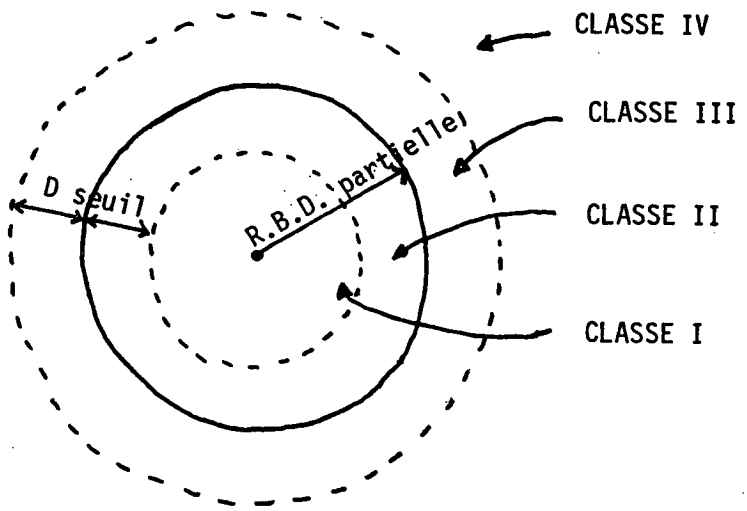


Figure 4 : Modélisation de la base de données partielle

CLASSE I : objets restant dans la B.D. partielle après déplacement du capteur

CLASSE II : objets susceptibles de sortir de la B.D. partielle

CLASSE III : objets susceptibles de rentrer dans la B.D. partielle

CLASSE IV : objets qui resteront à l'extérieur de la B.D. partielle.

Au cours du déplacement du capteur, tout objet appartenant à une classe est susceptible de passer dans une des classes qui lui est connexe.

A l'instant t_n, la base de données partielle est constituée par l'union des classes I et II cependant seuls les objets appartenant aux classes II et III pourront modifier le contenu de la B.D. partielle à l'instant t_{n+1}. Cette propriété, utilisée conjointement avec les propriétés topologiques définies précédemment, va permettre de gérer l'évolution de la base de données partielle en fonction du déplacement du capteur.

A l'instant initial t_o, on suppose le capteur immobile et les paramètres caractéristiques du capteur connus (le rayon R de la base de données partielle, les coordonnées (x_{c_o} , y_{c_o} , z_{c_o}) du centre de la B.D. partielle à l'instant t_o et le module de son vecteur de déplacement maximum Vd_{max}).

Le calcul de la distance caméra-objet permet d'affecter chaque objet de l'univers $\mathcal{U}$ à une des quatre classes définies précédemment. Le numéro de cette classe est alors mémorisé dans le pointeur S du fichier ITAB 1 (fig. 3).

Seuls les objets appartenant aux classes I, II et III interviennent explicitement dans l'algorithme de gestion de la B.D. partielle, ils sont regroupés dans un tableau dont la structure est représentée à la figure 5.

ITAB 2(IAD,J)

NUM OBJ si S = 1	NUM OBJ si S = 2	NUM OBJ si S = 3

Figure 5 : Structure du fichier ITAB 2

Le pointeur IAD permet d'assurer un chaînage entre les deux tableaux ITAB 1 et ITAB 2 (fig. 3 et 5).

Les deux premières colonnes de ITAB 2 représentent à l'instant t_n les objets contenus dans la base de données partielle V et la taille de ce tableau reste réduite et finie (proportionnelle à la densité d'objets dans l'univers $\mathcal{U}$) quelle que soit la taille de l'univers $\mathcal{U}$ qui peut contenir un grand nombre d'objets.

Enfin, un dernier tableau dont la structure est donnée à la figure 6 contiendra toutes les modifications apportées à la base de données partielle entre l'instant t_n et l'instant t_{n+1} (c'est-à-dire les objets disparaissant ou apparaissant dans cette B.D.).

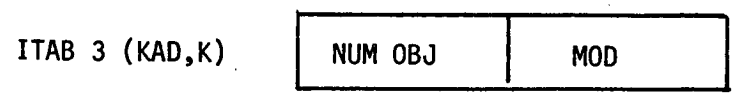


Figure 6 : Structure du fichier ITAB 3

Un récapitulatif des différents tableaux utilisés par l'algorithme de gestion dynamique, met en évidence les points suivants :

- la différence entre les tableaux entiers (associés aux pointeurs) et le tableau TAB 4 qui contient toutes les informations en format réel définissant les objets.

Dans un but d'optimisation temporelle, la majorité des opérations de l'algorithme portera sur les pointeurs.

- la hiérarchie entre les différents tableaux entiers.

* **ITAB 1** contenant tous les objets de l'univers $\mathcal{U}$ et génére hors ligne
* **ITAB 2** contenant tous les objets nécessaires à la gestion de la B.D. entre t_n et t_{n+1} (mis à jour dynamiquement)
* **ITAB 3** contenant uniquement les modifications de la B.D. entre t_n et t_{n+1}.

Pour gérer de façon dynamique les objets apparaissant et disparaissant du champ de vision du capteur lors de ses déplacements successifs, il est nécessaire de connaître à chaque instant l'évolution des différentes classes d'objets : les classes I et II qui constituent la base de données V à l'instant t_n

mais également les classes III et IV qui doivent être mises à jour de façon à être disponibles à l'instant t_{n+1}.

L'intérêt de l'algorithme proposé réside dans le fait que seules les transactions entre les différentes classes sont prises en compte en vue de limiter le nombre d'objets à traiter et donc, de minimiser les temps de calcul.

Ces transactions sont les suivantes :

Classe I <——> Classe II
Classe II <——> Classe III
Classe III <——> Classe IV

L'algorithme utilise les propriétés de connexité du graphe orienté représentant les relations de voisinage entre objets et dont les nœuds sont les objets et les arcs les directions des différents sous-espaces $\mathcal{E}_1, \ldots, \mathcal{E}_8$.

La recherche des objets dont l'état est modifié à l'instant t_{n+1} se fait par propagation au sein du graphe de connexité ; le critère d'arrêt étant l'état stable suivant :

- l'objet appartient à la classe IV
- son état n'a pas été modifié entre t_n et t_{n+1}.

La propagation dans le graphe est mémorisée dans le tableau ITAB 1 grâce aux pointeurs IP ("père" de l'objet en train d'être traité) et IS (qui pointe sur celui des huit sous-espaces "fils" correspondant au prochain objet à traiter). De même les modifications des classes associées aux objets se feront dans ce même tableau grâce à l'indicateur MS (0 : si pas de modification entre t_n et t_{n+1} ; 1 : si modification) et par la valeur de S indiquant le nouvel état de l'objet (fig. 3).

Parallèlement à la descente dans le graphe les tableaux ITAB 2 et ITAB 3 sont mis à jour en fonction des changements d'état des objets.

Les objets initialisant l'algorithme sont choisis dans une des classes II

et III car ce sont les seules contenant des objets susceptibles de modifier la B.D. partielle.

Cette B.D. partielle est déclarée valide à l'instant t_n lorsque tous les objets contenus dans ITAB 2 ont été traités.

La figure 7 résume dans le cas planaire (disque se déplaçant sur un plan) les diverses étapes de l'algorithme de gestion dynamique.

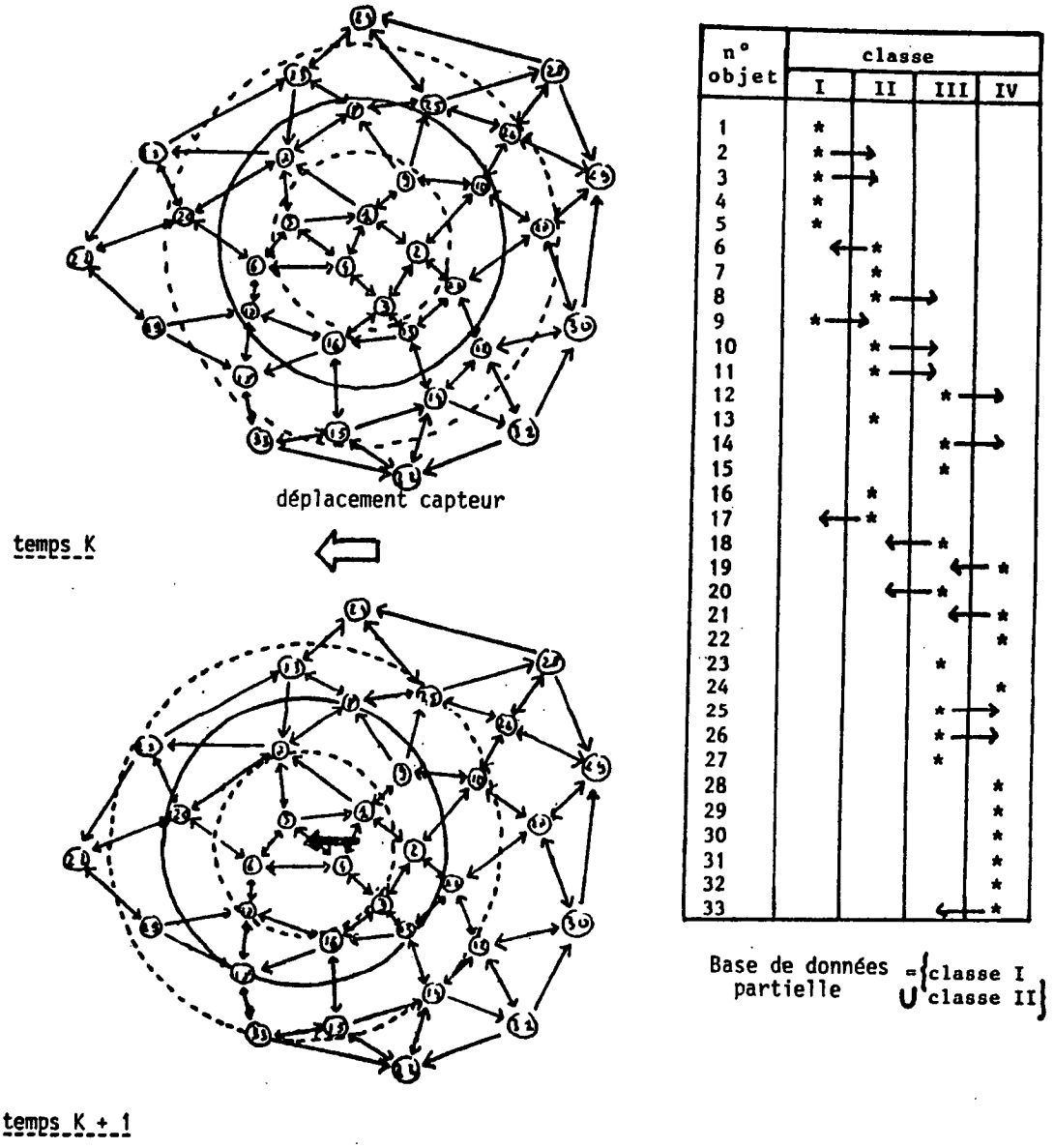


Figure 7 : Evolution de la base de données partielle en fonction du mouvement du capteur.

V.4 - Analyse des performances

Un programme de simulation a été écrit en FORTRAN afin de valider les performances de l'algorithme. L'usage de ce langage étant incompatible avec la vocation temps réel de l'algorithme, il semble plus juste d'évaluer les performances de l'algorithme au nombre d'objets traités lors de la mise à jour de la base de données partielle.

Le temps de traitement d'un objet est assimilé ici à celui du calcul de la distance caméra-objet, exécuté en flottant et faisant intervenir des élévations au carré.

La base de données globale décrivant l'univers $\mathcal{U}$ a été créée à l'aide d'un générateur aléatoire dont les tirs successifs représentent les coordonnées X_i, Y_i, Z_i des objets par rapport au repère univers $R_{\mathcal{U}}$. Les résultats présentés par la suite ont été obtenus avec un univers constitué par un cube de 10 mètres de côté et qui contient 1 000 objets distribués de façon uniforme.

Il est à noter que la taille de l'univers et le nombre d'objets ne sont limités que par la capacité de stockage du calculateur et qu'en aucun cas il n'influe sur le temps de calcul de l'algorithme. (A priori, même, l'usage d'un algorithme de gestion dynamique se justifie d'autant plus que la base de données globale augmente).

A partir de cette base de données, un programme établit (hors ligne) le graphe de connexité associé à l'univers $\mathcal{U}$ et construit le tableau ITAB 1 caractéristique de ce graphe.

Afin d'analyser les performances, un programme interactif a été développé qui permet :

- de définir les paramètres du capteur (portée, position initiale, module du déplacement maximum autorisé)

- d'initialiser la base de données partielle à l'instant t_o (initialisation de tableaux ITAB 1 et ITAB 2)

- de définir le déplacement du capteur entre l'instant t_n et t_{n+1} (module et direction du déplacement)

- d'activer l'algorithme de gestion dynamique de la base de données partielle.

Les résultats présentés dans le tableau 1 ont été obtenus en faisant décrire au capteur un circuit dans l'univers $\mathcal{U}$. Pour interpréter ces résultats, il est nécessaire de se reporter au modèle sphérique de la base de données partielle décrit précédemment à la figure 4. Un rapide calcul montre que, dans le cas d'une densité uniforme, le nombre d'objets trouvé dans chaque classe est bien directement proportionnel au volume de la classe et correspond à la valeur théorique.

D'un point de vue topologique, le travail de l'algorithme consiste à extraire du graphe associé à l'univers $\mathcal{U}$, un sous-graphe SG tel que ; à l'instant t_{n+1} :

- les nœuds internes du sous-graphe appartiennent à l'union des classes I, II et III.

- les nœuds terminaux de ce sous-graphe appartiennent à la classe IV et ils ont au moins un nœud connexe appartenant à l'une des trois autres classes.

En termes plus simples, cela revient à chercher l'enveloppe formée par les points de la classe IV qui sont le plus proche possible du volume balayé par la sphère S entre t_n et t_{n+1}. Le nombre total de points traités s'interprète comme la somme des points contenus dans ce volume et de ceux situés sur cette enveloppe.

La figure 8 montre le volume balayé par la sphère au cours d'un déplacement soit deux demi-sphères situées de part et d'autre d'un cylindre dont la hauteur est égale au déplacement du capteur entre t_n et t_{n+1}.

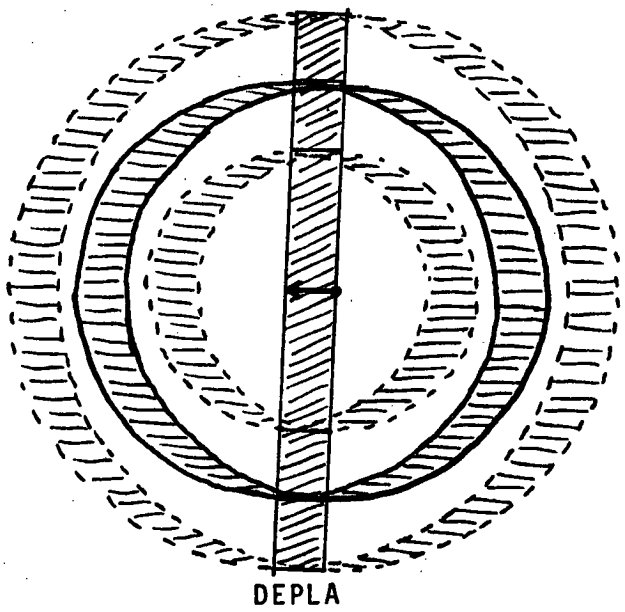


Figure 8 : Volume balayé par la B.D. partielle entre K et K + 1.

A partir de ce modèle et des résultats obtenus expérimentalement, il est possible de calculer la distance moyenne de l'enveloppe au volume balayé par la sphère entre t_n et t_{n+1}.

Pour une densité d'objets uniforme, le volume de l'enveloppe est donné par l'expression :

$$V = \frac{4}{3} \pi X^3 + \pi X^2 . |Vd|$$

avec

$$X = R + Vd_{max} + \Delta$$

R = rayon de la base de données partielle

Vd_{max} = amplitude maximale autorisée du déplacement du capteur

$|Vd|$ = module du déplacement entre t_n et t_{n+1}

Δ = valeur moyenne de la distance de l'enveloppe au volume balayé par la sphère.

Ce qui permet d'obtenir en introduisant la densité volumique P, le nombre de points contenus dans l'enveloppe.

$$N = \rho V = \pi\rho X^2 \left(\frac{4}{3} X + |Vd|\right)$$

Pour les résultats présentés au Tableau 1, la distance moyenne Δ a été trouvée égale à 0.9 mètre ce qui, compte tenu de la densité de l'univers (1 point/m^3), représente une pénétration très faible dans les points appartenant à la classe IV.

De plus, il est aisé de constater que l'accroissement du nombre d'objets traités (et donc du temps de calcul) dépend de la densité, du rayon de la base de données partielle et du module de déplacement maximal mais, est indépendant du nombre total d'objets dans l'univers global. Cette indépendance vis-à-vis de la taille de l'univers global met en évidence l'intérêt maximal de l'algorithme dans le case de larges bases de données.

L'algorithme de gestion dynamique utilise des notions telles que les pointeurs, les tests de drapeaux, les chaînages et la gestion dynamique des différents tableaux qui ne sont pas aisément manipulables avec des langages tel que FORTRAN, par exemple. On peut donc penser que les performances temporelles seraient fortement accrues en utilisant des langages plus proches de la machine tel l'assembleur.

V.5 - Conclusion

L'algorithme présenté permet d'extraire dans un univers 3D formé d'un grand nombre d'objets statiques décrit par un ensemble de leurs points caractéristiques, une base de données partielle constituée par les objets contenus dans une sphère de rayon paramétrable se déplaçant de façon aléatoire dans l'univers 3D.

La mise à jour de la base de données partielle se fait à chaque déplacement en minimisant le nombre d'objets nécessaires à l'algorithme grâce à l'utilisation d'un graphe de connexité des objets dans l'univers $\mathcal{U}$ qui a été établi dans une phase d'apprentissage. Cette particularité permet d'obtenir une indépendance du temps de calcul vis-à-vis du nombre d'objets de l'univers (à densité égale).

Les limitations de cet algorithme sont de deux sortes. Tout d'abord, nous avons considéré le passage de la représentation complète de l'objet vers une représentation simplifiée par un ensemble de points comme implicite.

Dans la pratique cette transformation est loin d'être évidente et nécessite une étude en soi. Le deuxième point concerne l'hypothèse de stationnarité de l'univers qui limite l'application, il sera nécessaire pour être réaliste de prendre en compte d'une façon ou d'une autre la présence d'objets mobiles et le mouvement de ces objets.

Instant	Nombre d'objets ITAB2 (B.D. partielle)			Nombre d'objets traités	Nombre de modifications de la B.D. partielle
	Classe 1	Classe 2	Classe 3		
K_0	1	3	9	100 (initialisat°)	0
K_0+1	0	5	6	48	1
K_0+2	0	3	8	43	2
K_0+3	0	4	7	43	1
K_0+4	1	3	9	42	4

TABLEAU 1 : Résultats de simulation.

VI. UTILISATION DU SYSTEME

PADL-2 est installé sur les VAX 11/750 YIN et YANG de l'IRISA (pour la connexion par le réseau faire C VAXI ou C VAXA respectivement).

Sur les deux VAX l'exécution du logiciel est lancée par la commande :

SYS $ UTLROOT : [MASPRO] PADL2

VI.1 - Exécution de PADL-2

Lors de l'exécution le système peut faire des recherches de fichiers (en lecture/écriture) dans six répertoires différents :

. le premier est le répertoire de travail courant;

. les cinq autres peuvent être assignés à PADL2 $ LIB1, PADL2 $ LIB2,..., PADL2 $ LIB5 avant l'appel au module exécutable qui se fait de la manière suivante :

RUN/NODEBUG SYS $ UTLROOT : [MASPRO.EXECUTE] PADL2

Actuellement l'exécution de PADL-2 est lancée à partir du fichier de commandes PADL2.COM situé dans SYS $ UTLROOT : [MASPRO], suivant le format:

@ PADL2 [NEWS] [DEB]

options: - NEWS : affiche le fichier PADL2.TXT situé dans [MASPRO.EXECUTE] (c'est l'option par défaut)

NONEWS : pas d'affichage

- DEB : exécute DEBUG (par défaut NODEBUG)

Ce programme de commandes permet en outre :

- d'assigner à PADL2 $ LIB3 le répertoire [MASPRO.EXAMPLES]
à PADL2 $ LIB4 le répertoire [PADL2.EXAMPLES.CHARS]
à PADL2 $ LIB5 le répertoire [PADL2.EXAMPLES.UTILITIES]

Dans ces 3 répertoires nous trouvons des exemples d'utilisation du langage de PADL-2 et des utilitaires qui permettent par exemple de visualiser un objet suivant quatre vues dont une en perspective et trois orthographiques comme suit (voir figure 5.1)

> USE <objet> ;

> LOGIN ; \ définitions des vues

> DISPO3R [<objet>] [: <set list>] ; \ dessin

LOGIN permet également de définir des noms de constantes, de couleurs, de terminaux et leurs valeurs associées.

- d'afficher les fichiers du type *.PFI contenus dans le répertoire de travail par la commande

$ DIR /VERSION = 1 *.PFI

- d'afficher le texte contenu dans PADL2.TXT

Chaque utilisateur peut donc créer son fichier de commandes à l'image de PADL2.COM en fonction de ses besoins et de l'organisation de ses données.

VI.2 - Intégration d'une application

Les différentes étapes à respecter pour réaliser l'édition de lieu avec PADL-2 sont les suivantes :

1. dans le répertoire SYS$UTLROOT : [PADL2] faire

   @ LOGIN

   @ 'TOOLS'ASSIGN

2. dans le répertoire SYS$UTL ROOT : [MASPRO] faire

   @ START

3. dans la commande LINK, ajouter à la liste des librairies dans lesquelles l'éditeur de lien fait ses recherches, le fichier PP2AP.OPT situé dans SYS$UTLROOT:[MASPRO] de la manière suivante

LINK file-spec[,...],SYS$UTLROOT:[MASPRO]PP2AP/OPTIONS

Pour lancer l'exécution de son programme d'application l'utilisateur pourra créer un fichier de commandes semblable à PADL2.COM.

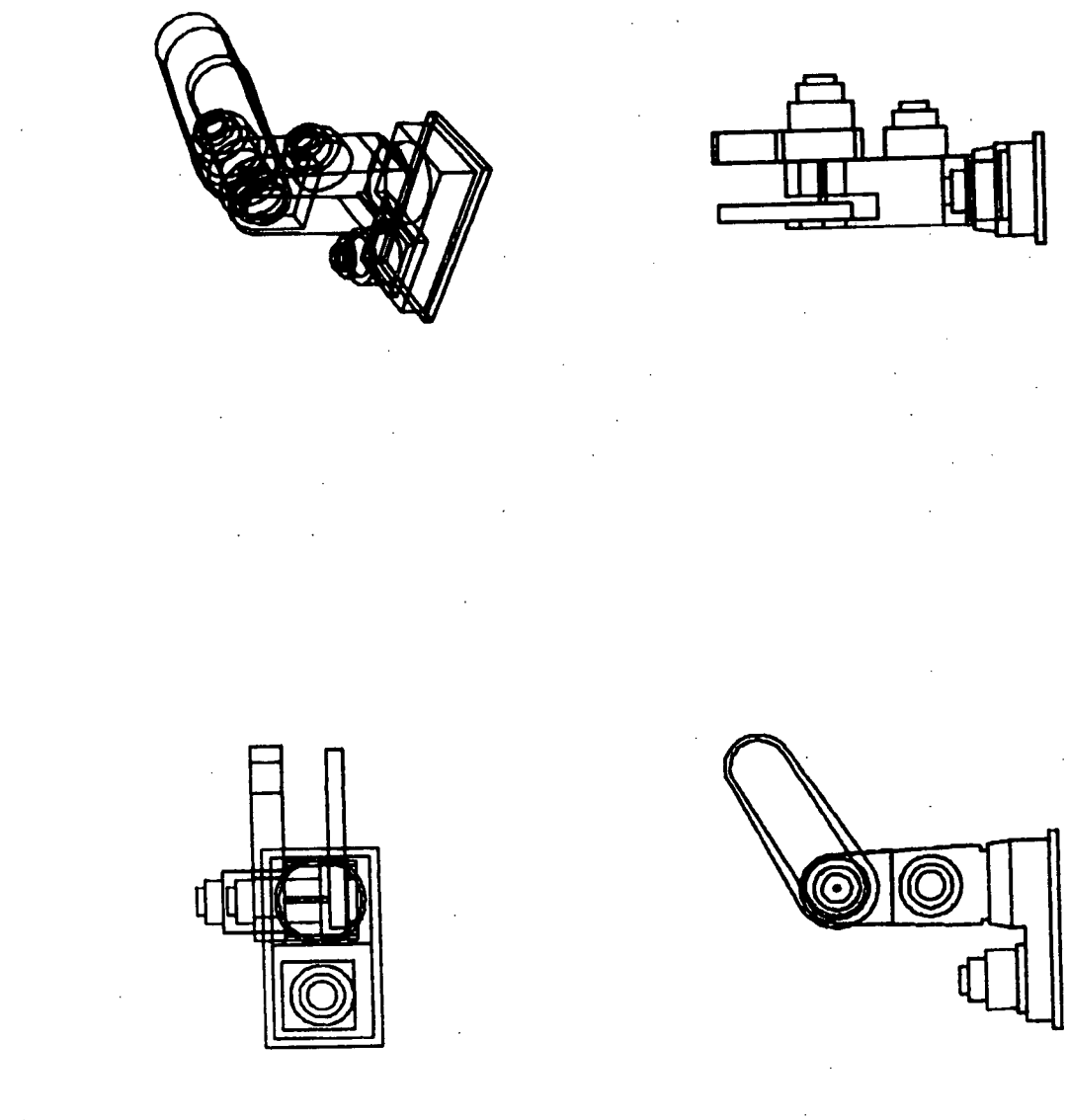


Figure 5.1: Visualisation d'un robot sous 4 vues.

BIBLIOGRAPHIE

[1] P. RIVES, L. MARCE : "use of moving vision sensors in robotics application to an obstacle ovoidance task". ICAR 1985.

[2] "Association de l'image et du mouvement en robotique dans un cadre 'boucle fermée' sur l'environnement local". Annexe technique Accord de collaboration "IRISA/THOMSON/LER".

[3] G. ANDRE : "Conception et modélisation de systèmes de perception proximétrique. Application à la commande en téléopération". Thèse de Docteur Ingénieur, Université de Rennes I, Octobre 1983.

[4] G. ANDRE, R. BOULIC : "Système graphique et capteurs proximétriques pour la programmation de robots". Proceedings of MICAD'85, Février 1986.

Nous donnons ci-dessous une liste non exhaustive d'articles généraux sur PADL-2 et la modélisation d'objets solides, rédigés par les membres du "Production Automation Project" de l'Université de Rochester, concepteurs du logiciel.

C.M. BROWN — PADL-2: A Technical Summary. **IEEE Computer Graphics & Applications**, vol.2, n°2, March 1982.

A.A.G. REQUICHA — Representations for Rigid Solids: Theory, Methods and Systems. **Computing Surveys**, vol.12, n°4, December 1980.

A.A.G. REQUICHA & H.B. VOELCKER — Solid Modeling: A Historical Summary and Contempary Assessment. **IEEE Computer Graphics & Applications**, vol.2, n°2, March 1982.

R.B. TILOVE — Set Membership Classification: A unified approach to geometric intersection problems. **IEEE Transactions on Computers**, vol.C-29, n°10, October 1980.

PI 264 — ***Constraints on long distance dependencies in grapping grammars***
Patrick Saint-Dizier – 20 pages ; Juillet 1985.

PI 265 — ***Automatice design of systolic chips***
Patrice Quinton et Pierrick Gachet – 21 pages ; Septembre 85.

PI 266 — ***Expression of syntactic and semantic features in logic-based grammars***
Patrick Saint-Dizier – 19 pages ; Septembre 85.

PI 267 — **Nestor : noyau d'exécutif pour le suivi en temps-réel des applications orientées robotique**
Philippe Belmans, Jean-Jacques Borrelly, Maryline Silly, Daniel Simon – 16 pages ; Septembre 85.

PI 268 — **Etude des méthodes de surveillance du comportement vibratoire des structures en mer : positionnement optimal des capteurs et détection d'anomalies**
Michèle Basseville – 70 pages ; Octobre 85.

PI 269 — **Modélisation et gestion d'univers 3D (une première approche à partir du logiciel PADL-2**
Gérard Hégron, Patrick Rives – 60 pages ; Octobre 85.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique